DRAFT

RECOMMENDED STANDARD APPLICATION SECURITY REQUIREMENTS

Version 2.0

11 March 2003

DEFENSE INFORMATION SYSTEMS AGENCY
Applications and Computing Security Division
Center for Information Assurance Applications
5275 Leesburg Pike
Falls Church, VA 22041

(This document is for review. Comments, if any, can be sent to JainD@ncr.disa.mil or KoehlerS@ncr.disa.mil)

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

Page

1. INTRODUCTION

1.1 Purpose

This document defines a set of recommended security requirements that are common to all software applications. The application security requirements identified in this document are intended to be used as a first step to designing security into applications to reduce application vulnerabilities. The general requirements will aid application developers in the identification and elimination of potential application vulnerabilities and security flaws proactively during the early phases of the lifecycle. Furthermore, this document contains security requirements specific to the Oracle database management system (DBMS) and Oracle applications that can be used to reduce development-related vulnerabilities.

This document also defines a "test objective" for each of the general requirements. The test objective is used to verify that the associated security measure is implemented within the application. These test objectives should help form the basis for the application's general test plan, and also provide input to the security test plan used for the application's Security Test and Evaluation (ST&E, see below). The test objectives can also be used to augment the security requirements and test objectives for the larger system to which the application may belong.

The requirements contained in this document include a compilation of existing DOD application requirements and a collection of security requirements that have been derived from security "best practices".

This document should be used as a reference during the phases of the DOD Information Technology Security Certification and Accreditation (C&A) Process (DITSCAP), i.e.:

- *Phase 1:* This document can be used to identify the application security requirements as you are identifying the system security criteria.
- *Phase 2:* This document should be referenced to conduct application security assessment as you are conducting the system certification analysis.
- *Phase 3:* Program managers should refer to this guidance when developing test plans and procedures for the ST&E of the application and/or the system of which it is a component..
- *Phase 4:* This document should be referenced to maintain the application security posture as you are managing the system security.

In addition, these requirements will be used in conjunction with Defense Information Systems Agency's (DISA) Security Technical Implementation Guides (STIGs) during the development cycle. The STIGs can be downloaded from the IASE (http://iase.disa.mil) or Field Security Operations (FSO) Guides (http://guides.ritchie.disa.mil) Web sites. Developing a STIG-compliant system and applying the recommended requirements listed in this document will ensure a higher level of security within an application.

Associated with this document, is the Application Security developer's guide developed to assist developers with implementing the requirements identified in this in Section 4. The third and fourth documents in this series identify application security assessment tools and present assessment methodologies that can be used to validate the test objectives presented in this document.

1.2 Scope

This is a "living" document." Application security requirements and test objectives will be refined as the document evolves. The "application specific" security requirements will also continue to be refined and added to as requirements for other applications are incorporated into the document.

This document identifies common vulnerabilities and develop requirements to reduce the occurrence of these vulnerabilities in applications. The vulnerability list is not all-inclusive and will be amended as required. This document does not contain vulnerability remediation information. Specific guidance on methods to avoid some of the general and specific vulnerabilities identified in this document are presented in the 'Application Security Developer's Guide associated with this document.

1.3 Intended Audience

Application developers should use this document as a guide for designing and implementing security features in their applications to run on DOD systems securely. The document will help application developers identify the application security controls to avoid creating vulnerabilities in their applications.

Furthermore, this document should assist application developers with creating well thought-out application designs that integrate security mechanisms early in the development lifecycle.

Some of the information in this document may also be beneficial to system administrators and system security engineers. However, system administrators and system security engineers interested in configuration and operational security requirements should refer to the various STIGs currently available from DISA.

1.4 Document Structure

This document consists of five sections and one appendix. An overview of the sections is provided below.

- *Section 1, Introduction:* Describes the objectives, scope, and structure of this document.
- *Section 2, Background:* Describes the DISA client's organizational mission, general application types, and the goals of application security.
- *Section 3, Application Vulnerabilities, Security Services, and Assurance Requirements:* Describes common applications vulnerabilities, security services performed by applications, and a discussion on criteria for determining security mechanism strength.

- *Section 4, Application Requirements:* Provides the security requirements and test objectives developed for general applications, grouped by the requirement categories defined in Section 3.

- *Appendix A, Acronyms and Abbreviations:* Lists the acronyms and abbreviations used throughout this document.

- *Appendix B, References:* Lists policy and guidance, and other documentation and online sources used to develop the application security requirements and test objectives provided in this document.

2. BACKGROUND

2.1 DISA's Role in Application Security

The mission of the DISA Application and Computing Security Division (Code API24) is to provide for the identification, development, system engineering, prototyping, provisioning, and implementation of various technologies supporting the defense-in-depth (DID) concept for multi-layered protection of the global applications and computing infrastructure of the global information grid (GIG).

The DISA Application and Computing Security Division has identified a set of application security requirements and common vulnerabilities for applications. It will identify these requirements in this document and define "test objectives" to provide DISA and other DOD agencies with application security guidance.

2.2 What is an Application?

An application is a software program or collection of software programs that execute on behalf of the operating system. An application uses the services of the computer's operating system and other supporting applications and is designed to perform a specific function directly for the user or, in some cases, for another application program.

For this document, the applications have been grouped into three main categories: server applications, client applications, and standalone applications:

1. Server applications-organize, retrieve, and/or transmit data at the request of a client application. Server applications include DBMS, Web servers, and directory server applications that organize, relate, store, manipulate, delete, retrieve data, and respond to user requests, made via client applications;

2. Client applications-request information from a server application for presentation to an end user, and to enable the end user to create, modify, and/or request storage or transmission of data by the server application;

3. Standalone applications-request storage and retrieval of data by the operating system's file system, to organize and present those data to the user, and to enable the user to create, modify, and/or transmit data.

The general requirements and test objectives developed in Section 4 will be applicable to some or all of the applications types described above.

2.3 Goal of Application Security

Application security, ultimately, provides the assurance that the application complies with and, when necessary, enforces, all security policies governing the application itself, the data it handles, the system to which it belongs, its operating environment, and its users. Application security has three primary objectives:

1. Ensure that the data an application creates, updates, stores, and/or transmits are protected from unauthorized disclosure, tampering, corruption, and destruction, by the application's users, by processes external to the application, and by the application itself.

2. Provide another security layer within the overall system, in accordance with DID strategy.

3. If required, provide security services that are not performed adequately or, in some cases, at all, by other parts of the system security infrastructure (e.g.; network, operating system, database management system security mechanisms, and any other security mechanisms within the overall system but external to the application). For example, if the access controls of the application's host are considered inadequate for protecting sensitive files created or modified by the application from disclosure, the application may include a programmatic interface to a cryptographic facility located in the surrounding infrastructure, enabling the application's "SAVE" process to invoke encryption each time it writes a file, thus causing the file to be stored in encrypted form instead of in the clear.

To achieve these objectives, the applications are responsible for ensuring that a core set of security services is performed. Applications can achieve the performance of security services via one of three ways:

1. *By performing certain security services itself:* for example, the application may need to perform its own logging of security-relevant events at the application level (vs. auditing at the operating system or DBMS level)

2. *By directly invoking security services to be performed by middleware or infrastructure security mechanisms:* for example, the application may include a programmatic interface or system call to a PKI to perform certificate validation in support of application identification and authentication (I&A) of users

3. *By verifying that certain security services have been performed by middleware or infrastructure security mechanisms:* for example, the application may be programmed to check for a flag or counter applied to a file by an external virus checker, to verify that the file has, in fact, been determined to be virus-free by that virus checker.

The following architecture diagram depicts the relationship of a typical software applications and its programmatic interfaces to its underlying host infrastructure and operating environment. The host infrastructure in this architecture incorporates the middleware and other supporting protocols not coded into the application itself, as well as the host platform (system software and hardware, including network and non-network device drivers and hardware).

The requirements in this document pertain to the software at the application layer of this diagram, and also to how that software interfaces with and relates to components at the lower layers.

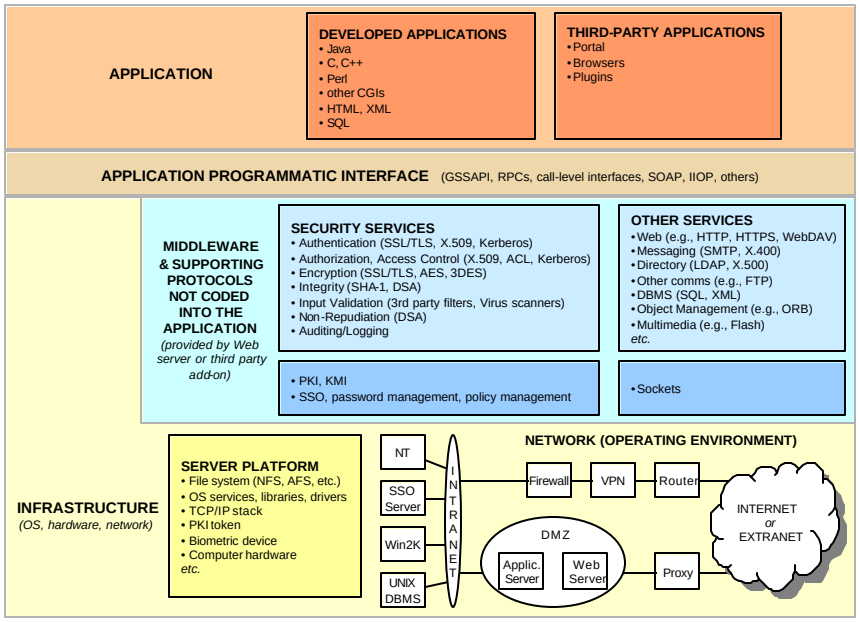

Figure 2-1. Typical Application Architecture

3. VULNERABILITIES, SECURITY SERVICES, AND ASSURANCE REQUIREMENTS

3.1 Application Vulnerabilities

The application vulnerabilities fall into three general categories:

1. *Design and development-related vulnerabilities:* Vulnerabilities that are caused (and can be prevented by) the designer and developer.

2. *Misconfiguration and administration-related vulnerabilities:* Errors that are introduced by administrators or operators when or after the application is installed in its operational environment.

3. *Necessary non-secure standards:* Vulnerabilities attributed to the implementation of a known non-secure protocol in order to perform a specific mission requirement. It may be necessary to utilize a known non-secure protocol to communicate with an older (legacy) system so that a required mission or business need can be performed. For example, SSL version 2 is known to be "broken" but if it is the only protocol supported by an obsolescent system, then mission need may override the risk.

The requirements in Section 4 of this document predominantly address vulnerabilities that fall into the first category (i.e., development-related vulnerabilities); a small number of requirements address vulnerabilities in the second category (i.e., misconfiguration/administration-related vulnerabilities, and specifically vulnerabilities caused during initial installation/deployment of the application). The third category of vulnerabilities (i.e., non-secure standards) is not addressed in this document, but is addressed to some extent throughout the Application Security Developer's Guide.

3.1.1 Common Vulnerabilities

Some of the most frequently reported vulnerabilities found in applications are identified below. The vulnerability codes will appear in the tables in Section 4, to identify the vulnerabilities that are addressed, in full or in part, by the security requirements in those tables.

Table 3-1. Common Application Vulnerabilities

Vulnerability	*Information About Vulnerability*
V1: Inadequate identification and authentication	Including authentication of users who should not be authenticated, or not requiring authentication at all.
V2: Insufficient access control	Restrictions on what authenticated users are not allowed to do are not properly enforced. Attackers can exploit these flaws to access other users' accounts, view sensitive data, or use unauthorized functions.
V3: Improper integration of application components	Integration that leaves "backdoors" or "security holes" that make it possible for users to bypass access controls, second-tier I&A, or other security controls, or to read security data passed between components. This includes incorrect interfaces between the application and any cryptographic mechanisms it relies upon.
V4: Weak passwords	Passwords that are too short, easy to guess, or not changed frequently enough.
V5: Plain text communication of sensitive information	Such as cleartext transmission of user passwords.
V6: Incorrect reparsing of data	Such as user-provided identification data passed between application and backend server.
V7: Susceptibility to buffer overflow	Application components (e.g., software libraries, drivers, application server components, CGI scripts, etc.) in C, C++, and some other languages do not properly limit input. Buffer overflow is caused by the developer's specification of a fixed or maximum (i.e., able to be exceeded) size of the data cache buffer allocated for storing data input into application program (e.g., by user through a forms interface); if the actual data exceeds the allotted buffer size, the excess data "spills over" from the data buffer into the processing cache (known as the "stack"), where it can cause denial of service (DoS) or, in some cases, be exploited by an attacker to take control of a process.
V8: Lack of adequate parameter validation	Information from user requests that is not validated before being used by the application can enable parameter tampering exploits by attackers, in which the attacker targets backend components through the application.
V9: Input validation of data containing active content	Can cause the active content to execute, and make the application vulnerable to "cross site scripting." In cross site scripting attacks, the application can be used to transport an attack to an end user's browser, allowing the attacker to view the user's session token, attack the user's workstation, or spoof content to fool the user into responding to the Web server in a way that the user's doesn't expect or intend.
V10: Acceptance of meta code embedded within input data	Enables "stealth commanding," i.e., the insertion of shell metacharacters in data input—e.g., "!" (which is used to access the command history in some shells; particularly troublesome in tcsh, where "!" can be used not just interactively, but in scripts) and "|" (the "pipe") in Perl. Many Perl programs allow the user to input a filename, and then pass that filename to a program via a shell command. However, because the shell may interpret characters differently than the Perl program, if the user includes "|" within the filename, the shell will attempt to execute the rest of the filename as a program; this vulnerability enables a malicious users to designate an ostensible filename containing the "|" followed by a complete control string as the "rest of the filename" in order to trigger hidden debug code or other developer backdoors left in deployed code (another vulnerability). Similarly, metacharacters and other special characters may enable a malicious user to insert entire programs in the application's data input fields, a technique called "cross site scripting."
V11: Acceptance of illegal characters in structured query language queries	A problem with SQL queries embedded within input data, which can cause the execution of spurious database commands.
V12: Use of relative pathnames	Enables users to gather information about the directory structure and content—information that can be used to launch other types of attacks.
V13: Acceptance of truncated pathnames	If no filename is specified at the end of the pathname, the system may simply list the full directory contents to the user, enabling the hacker to gather information about the

	directory structure and content.

Vulnerability	*Information About Vulnerability*
V14: Links to pathnames no longer present on the server	Such links may reroute the user to the Parent Directory, or even the "root" directory, enabling the user to gather information about the directory structure and content.
V15: Inadequate or inefficient error or exception handling or recovery	Error conditions that occur during normal operation are not handled properly by the application. If an attacker can cause an error that the application is unable to handle, the attacker can obtain detailed system information, cause denial of service to the application security mechanisms, the entire application, or the server on which the application runs. Furthermore, the application does not fail safe, making resources vulnerable to attack during/after failure. The application does not ensure restoration to a secure state after restart. The application enables attackers to flood the application with malformed arguments, overwhelming the exception/error handling facility, and causing denial of service.
V16: Common Gateway Interface (CGI) script "holes"	CGI scripts can present security "holes" in two ways: (1) They may intentionally or unintentionally leak information about the host system that will help hackers break in. (2) Scripts that process remote user input, such as the contents of a form or a "searchable index" command, may be vulnerable to attacks in which the remote user tricks them into executing commands. These "holes" be exploited to compromise applications that run as "root," or even to exploit applications that run with minimal privileges, to gain direct access to the underlying OS.
V17: Presence of developer backdoors	The application code has not been adequately debugged, "cleaned up", tested, or certified before going into production. "Clean up" entails removal of debugging accounts, debugging passwords, debugging tools, debugging and testing flags, and other developer "backdoors" from the application code and operating environment before operational deployment. Such backdoors often grant the user higher (developer-level) privileges on the system, and can be exploited by attackers who discover them in deployed applications.
V18: Password grabbing and replay	A problem when passwords are transmitted in the clear and/or over an unencrypted link.
V19: Susceptibility of cookies to content changes	Because of inadequate security protections in, cookies are particularly vulnerable to alteration ("cookie poisoning")—for example, to change the user identity information in the cookie, enabling a malicious user to use the cookie to impersonate a valid user.
V20: Lack of access controls on directly-typed Uniform Resource Locators (URL)	A problem if access controls are implemented at the application level, and expect all access attempts to be made by clicking a link on the portal or Web page because the access controls will not be invoked if the URL is directly typed into the browser's "Location" line instead.
V21: Use of hidden fields	If a hidden field is used in a Web page without validating the source of Hypertext Markup Language (HTML) updates to the field, the page will be susceptible to "hidden manipulation," a kind of replay attack in which the user captures the HTML source, changes the content of the hidden field, then plays the modified HTML source back to the server, which will accept the new hidden field value. This method is most often used by malicious users to lower (to zero) the prices of expensive items sold via E-commerce. Similarly, a CGI script could have its parameters modified to search not for the hidden field containing a price code, but instead for a hidden password file.
V22: Susceptibility to Web page "defacement"	The Web pages stored and accessed by the application can be modified or replaced by unauthorized users, due to V1, V2, V34, V35

Vulnerability	*Information About Vulnerability*
V23: Incorrect interoperation with surrounding security infrastructure	The application's interfaces to its surrounding security infrastructure, e.g., its invocation of PKI components, are insecure or contain errors, thus providing "holes" through which attackers can compromise both the application and the infrastructure.
V24: Insufficient or alterable tracking and recording of user actions	The application does not adequately detect and/or log security-sensitive functions performed by its users, making it difficult to do a forensic analysis of a violation that may have been instigated through use of the application. Also, the application may not protect the integrity (against modification) and availability (against deletion) of its logs.
V25: Presence of unnecessary system calls, processes, libraries, data types, etc.	The application code contains processes, calls, data types, etc. that are never invoked or used by the application. These calls, if detected by an attacker, can be used to cause the application to behave in unexpected ways.
V26: Granting of excessive or unnecessary privileges	Users or application processes are granted privileges that exceed their authorization rights, or that are unnecessary for them to perform the operations they are, according to their role, group, or individual identity, supposed to perform. This problem can occur at two levels: authorization of users that grants them excessive privileges, while authorization of processes within the application can cause them to demand more privileges than the user on whose behalf they operate is entitled to.
V27: Resource conflicts make application susceptible to subversion or failure	Different processes in the application attempt to access the same resource, e.g., to write to the same memory address, simultaneously, causing a conflict that may lead to a suspension of processing that may be exploited by an attacker.
V28: Unnecessary complexity	The application is unnecessarily complex—the code is poorly structured and internally inconsistent, contains lots of GoTos and recursions, making it difficult to trace a single function from initiation to termination; it contains long multifunction modules instead of short single-function modules; it contains processes with multiple entry and exit points; it attempts to perform too many complicated or seldom-needed functions. In all cases, the complexity makes the application very difficult to certify and accredit, and is much more likely to contain undetectable vulnerabilities that can be found and exploited by attackers.
V29: Insecure installation or configuration	Failure to assign the most restrictive host access controls possible to the directories containing the application executables, files used by the applications, and data created by the application that will still allow the application operate correctly. Failure to remove developer "backdoors" (debug accounts and functions), and to change default passwords, etc.
V30: Presence of insecure (non-certified) third-party components	The application has been integrated to contain software and/or hardware components that have not been certified under NIAP to an Evaluation Assurance Level sufficient to protect the application and its data from the risks of its particular operational situation and environment. Because modifying the source code of non-Open Source third-party components is usually not viable, determining how to fix vulnerabilities in these components, and in their interfaces with the rest of the application, is difficult and usually requires full cooperation of the vendor, which is often not forthcoming with larger corporations.
V31: Use of insecure system calls, processes, services, libraries, data types, etc.	The application code uses system calls, processes, software libraries, services, etc. that contain known vulnerabilities or represent known threats to the application's security, for example HTTP instead of HTTPS for transmission of sensitive Web data; C/C++ libraries that lack input validation and thus introduce buffer overflow vulnerabilities; system calls that can cause buffer overflow situations, etc.

Vulnerability	*Information About Vulnerability*
V32: Lack of protection from malicious code	The application's host and surrounding infrastructure do not adequately scan for viruses and other malicious code, and the application itself provides no interface to a virus scanning facility. In the case of mobile code applications, the application provides no sandboxing mechanism.
V33: Persistent processes, sessions, connections susceptible to hijacking	The application fails to terminate processes, network connections, or user sessions within a configured timeout, and instead allows them to remain active indefinitely; or the configured timeout is too long.
V34: Inadequate protection of data in transit from tampering	The application does not invoke an integrity mechanism such as a digital signature or message digest hash and affix that mechanism to data before transmitting them.
V35: Inadequate account and session management	Account credentials and session tokens are not properly protected. Attackers can compromise passwords, keys, session tokens, or other tokens to defeat authentication restrictions and assume other users' identities.
V36: Command injection	When the application passes parameters, or SQL commands, as they access an external system—e.g., database, operating system, etc.—the attacker may be able to embed malicious commands in the passed parameters, causing the external system, which does not suspect the sabotage of the parameters, to execute those commands.
V37: Insecure remote administration	Many Web applications allow administrators to access the Web server using a browser interface. Unless the browser-server session is strongly protected (e.g., using SSL, X.509 certificates, etc.), an attacker may hijack the session or intercept the sensitive data flowing between browser and server, in order to inject spurious commands, tamper with the data, or disclose sensitive information that can be used to help mount an attack on the application.
V38: Multiple programming languages	The application is written in a variety of non-interoperating languages, making it very difficult to abstract security-related code behind a clean API. Security-related code is scattered throughout the application, resulting in multiple replications of the same vulnerabilities throughout the application.
V39: Use of programming language(s) not conducive to development of secure applications	The language used for developing the application is not conducive to writing security-related code. This is particularly true of languages used in Web applications, such as Pre-Hypertext Processor (PHP) and Microsoft Visual Basic Scripting Edition (VBScript) are untyped, and thus make it very difficult to predict their compile-time behavior.

3.1.2 Causes of Vulnerabilities

Development-related vulnerabilities are typically introduced into applications in one of the following ways:

- *Insufficient security requirements:* The application's security requirements are incomplete or poorly defined;

- *Weak design:* The overall application design does not incorporate security effectively. For example, a poorly designed session-handling mechanism in a Web application may allow users to manipulate cookies in a way that enables them to bypass authentication controls and change accounts at will;

- *Implementation errors:* Coding errors and other "bugs" that can be exploited by hackers to alter application functionality or to run arbitrary commands. Implementation errors include buffer overflows, format string errors, and race conditions. Implementation errors make up the majority of reported vulnerabilities in commercial off-the-shelf (COTS) applications;

- *Malicious code:* Back doors, logic bombs, and salami code are examples of malicious code inserted into code by programmers to be later exploited by them for their own ends;

- *Deployment errors:* Errors that result from improperly deploying the application into production. Includes making incorrect assumptions about the deployment environment in which the application will operate, failure to remove debugging accounts and passwords, and failure to accurately maintain version control;

- *Inadequate quality assurance and testing:* Unless security is incorporated into quality assurance (QA) and testing of applications, the vulnerabilities described above, such as design flaws and buffer overflows, will not be caught during testing. Regular software testing should be expanded to include not only tests against normal, expected data sets, but also tests that subject the application to typical hacker attacks.

3.1.3 Discovering Application Vulnerabilities

Although some first-generation application vulnerability scanning tools that check for known vulnerabilities in applications have come on the market, the most effective way of eliminating most application vulnerabilities remains a thorough, expert analysis of the application source code. Numerous vendor and open source code scanning tools also are available. Some of these same tools are, in fact, tools used by hackers to scan applications to gather information about application source code—including Hypertext Markup Language (HTML), JavaScript, and particularly browser source code (which is not private)—to discover the application's exploitable vulnerabilities.

3.2 Application Security Services

The following sections identify security services that may be performed or invoked by an application and define the security service as it relates to application security requirements presented in this documents.

3.2.1 Identification and Authentication

Application identification and authentication (I&A) is a two-stage process designed to ensure that the user that is attempting to use the application to perform some function is provably known to the application. The first stage of I&A is identification—i.e., the recognition by the application of a user's identity (or of the identity of a process acting on a user's behalf). This identification of the user is most often achieved through the use of a unique machine-readable name associated with the user. The second stage of I&A is authentication, whereby the application verifies that the user's identity does, indeed, belong to the user who claims it. Authentication is achieved by validating a trustworthy credential associated with the user's machine-readable name. This credential may be a password (static or dynamic), a digital certificate, or a biometric. I&A is the

basis for all other security in the system because I&A establishes the trust relationship between the user and the application.

3.2.2 Authorization

Application authorization is the process whereby an authenticated user's (or process') identity is associated with (bound to) the rights and privileges that will govern his ability to access data using the application, and to perform particular application functions. Authorization is usually implemented through one or more of the following:

- Binding of an individual user to a role (i.e., Role-Based Access Control [RBAC]) or a user group, whereby the user "inherits" the privileges, access rights, and functions associated with that role or group,
- Assignment of Discretionary Access Control (DAC) permissions to the user,
- Inclusion of the user in an access control list (ACL),
- Inclusion of authorization extensions in the user's X.509 certificate,
- Assignment of an American National Standards Institute (ANSI) X9.57 attribute certificate to the user.

3.2.3 Access Control

Application access control is closely related to application authorization and ensures that access to the application's data and functions is permitted only according to the permissions explicitly stated in the user's authorizations, and that no access is allowed by an unauthorized user or by an authorized user in violation of the user's authorizations.

3.2.4 Confidentiality

Application confidentiality is the assurance that the information created and used by the application (including information pertaining to network addresses, identities of users) cannot and will not be disclosed to unauthorized persons. Applications are responsible for ensuring the confidentiality of information they store and information they transmit. Confidentiality is usually implemented through one of the following:

- Invocation of an infrastructure cryptographic mechanism to encrypt the data and/or the network connection over which the data are transmitted, with only authorized parties having access to the keys required for decryption,
- Binding of information labels and markings to data created and modified using the application,
- Assurance that object reuse has been enforced, including deletion of temporary data, cache, and files created by the application program during execution.

3.2.5 Integrity

Application integrity is the assurance that the information, executable processes, and resources that constitute the application and those used by the application can and will be protected from unauthorized, unanticipated, or unintentional corruption or modification. Applications that are used to create and modify data are often responsible for ensuring that the data are accurate.

3.2.6 Availability

Application availability is the assurance that the data, executable processes, and system resources that constitute the application and those used by the application will be in place (in the case of executables, operational) when the user (or process) needs them and will be in the form needed by the user.

3.2.7 Accountability

Application accountability is the assurance that all activities and transactions performed through the application by users (or processes acting on behalf of users) are traced back to those users, in order to hold them for those activities/transactions. The most common mechanism for implementing accountability is auditing. In many application environments, audit mechanisms are provided only at the operating system and database management system (DBMS) levels. For this reason, application audit must be achieved by capturing data through the application's event logging or error logging mechanism, which may have to be modified in order to allow for capturing of security-related events—and information about those events—that would not otherwise be captured by the logging mechanism.

3.2.8 Non-Repudiation

Application non-repudiation is the irrefutable, provable association of the identity of a user (or process operating on a user's behalf) with a piece of information (datum) that the user used the application to create, modify, delete, transmit, or receive. This information may be a file or message, the user's security information (e.g., authorizations) used by the application, or the application's own configuration data.

3.3 Assurance of Application Security Mechanisms

In addition to defining a requirement for a given application security mechanism, it is important to determine the optimal and minimal acceptable strength—assurance—of that mechanism given the risk assessment of the application and its operating environment. This risk assessment will determine the vulnerability of the operational environment and of the application itself, based in part on its purpose, criticality, and functionality, and given these factors the susceptibility to threats.

Significant factors in determining the application's susceptibility to threats are the sensitivity and classification of data to be handled by the application and the clearances and/or other authorizations of the application's end users. The more sensitive the data, the higher value it represents as a potential target, the stronger it needs to be protected from attack.

What follows is a high-level overview of the factors to be considered when determining the appropriate level of assurance for application security mechanisms. Refer to Department of Defense Instruction (DODI) 8500.2 (6 February 2003), "Information Assurance (IA) Implementation", E.4 Enclosure 4 "Baseline Information Assurance Levels" for a more comprehensive discussion of this material.

Further guidance that may also be helpful, based on the correlation of Common Criteria Evaluation Assurance Levels and Strength of Mechanism Levels with Office of Secretary of Defense Global Information Grid Information Assurance Policy-defined Robustness Levels, is found in the NSA Information Assurance Technical Framework (IATF), in Section 4.5 ("Robustness Strategy"), with further discussion of Robustness Levels provided in Appendix E of the IATF. These discussions are clear and concise and provide practical examples.

3.3.1 Mission Assurance Categories

In DOD, the criticality and risk profile of applications (and their associated data) are expressed in terms of Mission Assurance Category. The lower the number of the Mission Assurance Category, the more critical the application and the more important that its security mechanisms and functionalities be robust in terms of assurance against potential threats to confidentiality, integrity, and availability.

Mission Assurance Categories express the application's mission criticality and associated characteristics, based on its purpose and its user community. DOD has defined three Mission Assurance Categories for characterizing DOD systems and applications. Table 3-2 presents the Mission Assurance Categories as defined in DOD Directive (DODD) 8500.1, Information Assurance (1 February 2002), DODI 8500.2, Information Assurance (IA) Implementation (6 February 2003), and the Assistant Secretary of Defense memorandum Department of Defense Public Key Infrastructure (PKI) (12 August 2000).

The application's Mission Assurance Category should be used as a criterion when determining, in particular, the necessary assurance of the application's availability, including the availability of its security functions—whether those functions are incorporated into the application itself, or are provided by external mechanisms/processes and called by the application. A MAC I application will necessarily require a stronger assurance of availability, and thus should be designed to incorporate availability measures and countermeasures, such as internal redundancies, robust error/exception handling, fault tolerance, etc. that will not necessarily be required for a MAC III application.

Table 3-2. Mission Assurance Categories

Category	*Characteristics of Data*	*Characteristics of Systems*	*Old Designation*
MAC I	Vital to operational readiness or mission effectiveness of deployed and contingency forces. Absolutely accurate, timely, available on demand. Classified, sensitive, or unclassified.	National Security Systems*, including systems used to directly perform: Intelligence activities, cryptologic activities related to national security, command and control of military forces integral to weapon or weapons system. Also other system directly critical to military or intelligence missions.	Mission critical
MAC II	Important to support of deployed and contingency forces. Absolutely accurate. Can sustain minimal delay without serious effect on operational readiness or mission effectiveness. May be classified; more likely Sensitive But Unclassified (SBU) or unclassified.	Identified by combatant commanders (CDRs): systems which, if not functional, would preclude the CDR from conducting missions across all operations, including Readiness, Transport, Sustainment, Modernization, Surveillance/reconnaissance, Finance/Contracting, Security, Safety, Health, Information Warfare, Information Security	Mission support
MAC III	Necessary to conduct day-to-day business. No material short term effect on support to deployed/contingency forces. May be classified; much more likely SBU or unclassified. Required to perform department-level and component-level core functions.	Administrative	n/a

** As per Clinger/Cohen Act, Title 10 of the U.S. Code, Section 2.3.10*

3.3.2 Sensitivity Levels

In addition to the application's Mission Assurance Category, the sensitivity of the data the application handles should be considered when determining the type and assurance level of security services the application must provide or obtain from its underlying host/infrastructure. An application that is used to handle data of a certain classification level will likely have different requirements for the type and assurance level of the confidentiality services it provides (or calls out to) than an application that is used to handle sensitive-but-unclassified (SBU) data, or publicly-releasable data. In addition, the non-hierarchical nature of the data—whether it is compartmented, caveated, categorized, or SAMI—may also affect the types and assurance levels of the application's required confidentiality services. While there will be some confidentiality requirements that must be met by all applications, regardless of the sensitivity of the data they handle, the technology used to satisfy those requirements, and the assurance level of that technology, will likely vary depending on the sensitivity of the data to be handled.

3.3.3 Levels of Concern and Levels of Robustness

The third set of factors to consider in determining security requirements are those associated with the formalized expression of perceived risk and the formalized expression of the strength of countermeasures to be employed to counteract that risk, specifically in terms of the strength of Information Assurance (IA) solutions and services used as anti-risk countermeasures.

The formalized expression of risk is the Level of Concern. The Level of Concern may be associated with the entire environment in which an application will operate, or it may be associated with the specific system to which the application belongs. In either case, Level of Concern expresses the potential vulnerability of the system or environment that is created by the perceived threats posed to it.

The formalized expression of countermeasure strength is the Level of Robustness. As with Level of Concern, Level of Robustness may be associated with an entire operating environment, or with a specific system. Level of Robustness is directly derived from Level of Concern: the higher the Level of Concern associated with an environment or system, the higher the Level of Robustness its security mechanisms much achieve in order to counteract the level risk expressed as Level of Concern.

There is also a direct correlation between Mission Assurance Category and Levels of Concern and Robustness. For example, MAC I applications are always considered to have a high Level of Concern and to require a high Level of Robustness.

The discussion of DID in the GIG Information Assurance Implementation Guide (Assistant Secretary of Defense for Command, Control, Communications & Intelligence [ASD C3I] DOD Chief Information Officer [CIO] Memorandum 6-8510) specifies three possible Levels of Robustness for the security services provided by technical information assurance (IA) solutions: high, medium, or basic. As noted, Levels of Concern and Levels of Robustness apply not to just individual IA services, but to entire operating environments in which those services collectively provide IA. However, for purposes of application security, Level of Concern and Level of Robustness should be considered both in relation to the overall requirements of the application—an high Level of Concern application will require a high Level of Robustness in terms of the quantity and assurance of the countermeasures it provides—and in relation to the requirements for assurance and functionality of individual security functions performed (or called) by the application.

For example, in a low Level of Concern environment, the application will need to have only a low Level of Robustness—in practical terms, this means that its security mechanisms can have a low level of assurance, and that certain mechanisms may not even be required. In terms of specific mechanisms, this may mean that Web client application may not need to invoke SSL to encrypt an HTML form before transmitting it over a network to the Web server application; in a medium Level of Concern environment, the application may need to invoke SSL encryption of HTML forms, but the network itself may not be. In a high Level of Concern environment, it may be necessary to use Type 1 encryption to encrypt the network over which the application must transmit the SSL-encrypted HTML form..

The definitions of each Level of Concern and Level of Robustness from the GIG IA document are listed in Table 3-3.

Table 3-3. Levels of Concern and Levels of Robustness

Level	*Level of Concern Definition*	*Level of Requirement Definition*	*Mission Assurance Category*
High	Information requires the most stringent protection measures and rigorous countermeasures	Security services and mechanisms provide the most stringent protection measures and rigorous countermeasures	I or II
Medium	Information requires layering of additional safeguards above the DOD minimum (basic) standard safeguards	Security services and mechanisms provide for layering of additional safeguards above the DOD-defined minimum ("basic") standard of safeguards	II
Basic	Information requires implementation of DOD-defined minimum standard of safeguards	Security services and mechanisms equate to good commercial practices	III

Appendix E of the Information Assurance Technical Framework (IATF) Version 3.0, a document explicitly called out by the GIG Policy (DOD CIO Memo 6-8510), should be used as a reference document when determining which Office of the Secretary of Defense (OSD)-defined Robustness Level to be associated with a particular environment/system. Section 4.5 of the IATF defines the specific cryptographic technologies, key lengths, etc. that should be employed at each Robustness Level. For applications that incorporate or invoke encryption, this guidance should be useful when defining the requirements (in terms of overall assurance and specific implementation details) of the cryptographic solution to be used by the application.

3.3.4 Strength of Cryptography

In practical terms, cryptography is likely to be used to implement some if not all of the application's security services. DOD policy and NIST Federal Information Processing Standards (FIPS) provide extensive direction that should help developers determine the specific characteristics of the cryptographic technology that should be used to implement these application security services.

As with other security mechanisms, the characteristics of cryptography are determined, to a great extent, by their required Level of Robustness which is also expressed as a Strength of Mechanism Level (SML), and may also be expressed in terms of a Common Criteria Evaluation Assurance Level (EAL). IATF Release 3.0 Section 4-5 and Appendix E are good sources of guidance for determining the SML and EAL that a cryptographic solution should have, given the required Level of Robustness indicated by the application's Level of Concern. Some DOD applications' Level of Robustness may allow them to use National Institute of Standards and Technology (NIST)-approved (vs. National Security Agency (NSA)-approved) cryptographic technologies, as described in FIPS 140-1 and 140-2, "Cryptographic Module Validation Lists," and Computer Systems Laboratory (CSL) Bulletin FIPS 140-1, "A Framework for Cryptographic Standards." These documents delineate the characteristics of four increasingly secure levels of cryptographic technologies, and provide a good basis for determining the appropriate SML and EAL of NIST-approved cryptographic technology to be used in candidate applications. These determinations should be made based on the results of a thorough and accurate risk assessment.

The reason it is important for application developers to understand the various requirements for cryptography, even if they plan to provide only APIs to DOD PKI or another DOD-approved cryptographic technology, is that this understanding will help ensure that their applications are not implemented in a way that could preclude ‘ the application’s ability to accommodate planned DOD migrations to more robust, or simply different, cryptographic technologies.

The sensitivity/classification of information, and the protection provided by the network over which that information will be transmitted, are also important factors in determining the necessary Level of Robustness of the cryptographic implementation—and specifically the cryptographic certificates—to be used to protect the information. DOD PKI Policy indicates the certificate class to be used for classified data (in all Mission Assurance Categories), unclassified data in MAC I, and unclassified data in MAC II and MAC III. The relevant NSA and NIST policies indicate other cryptographic strength characteristics to be considered, such as key lengths, algorithms, and standards to be used in conjunction with data at different sensitivity/classification levels.

3.3.5 X.509 Certificate Assurance Levels

The X.509 Certificate Policy for the United States Department of Defense, Version 6.0 (31 May 2002) provides clear guidance on the assurance levels of X.509 certificates to be used in DOD systems (defined in Section 1.3.4.6 of the Policy). This guidance is further clarified in Department of Defense (DOD) Class 3 Public Key Infrastructure (PKI) Public Key-Enabled Application Requirements (13 July 2000), Section 3.2, “PKI Assurance Levels”.

The assurance level of a certificate is expressed in terms of the certificate’s class. ASD C3I Memorandum, Department of Defense (DOD) Public Key Infrastructure (PKI) 12 August 2000 (known as the DOD PKI Policy) provides direction on determining the appropriate level of assurance for DOD PKI certificates used in DOD systems (page 2 of the DOD PKI Policy, “Selection of Appropriate DOD PKI Certificate Assurance Levels”). (COMMENT: Simplify last sentence.)

The X.509 Certificate Policy, while consistent with the DOD PKI Policy, is not limited to DOD PKI certificates, and is intended to apply to all types of certificates used in DOD systems, including the FORTEZZA® certificates used in the Defense Message System (DMS). The guidance in the X.509 Certificate Policy defines the assurance level (class) of certificate that should be used given the combination of the Value of the Information (defined in Section 1.3.4.3 of the X.509 Certificate Policy; to some extent comparable with Mission Assurance Category) to be protected, and the Level of Protection of the Network Environment (defined in Section 1.3.4.5 of the X.509 Certificate Policy; to some extent comparable to Level of Robustness) in which that information will be used. Guidance for General Usage of DOD Certificates appears in Section 1.3.4.6, with a Summary of that General Usage guidance in Section 1.3.4.7.

The information in Section 1.3.4 of the X.509 Certificate Policy implies that the combinations of Value of Information and Protection Level of Network Environment can be expressed more simply in terms of Mission Assurance Category.

4. APPLICATION SECURITY REQUIREMENTS

Section 4 provides application security requirements grouped into the security service categories described in Section 3.2, as well as requirements in areas not specific to a particular security service, such as application interaction with the host environment, general use of cryptography, design and coding, and preparation for deployment. The requirements in Section 4 are not meant to provide an exhaustive set of security requirements for all applications, but instead present a minimum set of requirements that should be satisfied by the majority of applications—noting that certain requirements will be applicable only for to certain types of applications, as indicated in the "Assumptions and Constraints" entries for certain requirements throughout the tables in Section 4.

These application security requirements are presented in a series of tables, provided within the subsections of Section 4. Each table lists and provides information on requirements pertaining to a particular application security objective or a security service to be performed by the application. These subsections and their tables include:

- *Subsection 4.3, Application Interaction with Underlying Host:* Defines how the application should securely interact with its underlying host environment.
- *Subsection 4.4, General Use of Cryptography:* Defines how the application should use cryptography in general (vs. cryptography to implement a particular security service).
- *Subsection 4. 5, Design and Coding:* Defines requirements for the application's design and coding to ensure its security.
- *Subsection 4.6, Identification and Authentication (I&A):* Defines how the application should implement I&A of users and processes.
- *Subsection 4.7, Authorization and Session Control:* Defines how the application should implement authorization of users and processes.
- *Subsection 4.8, Access Control:* Defines how the application should implement access control on its resources.
- *Subsection 4.9, Confidentiality:* Defines how the application should ensure the confidentiality of the data it handles and uses.
- *Subsection 4.10, Integrity:* Defines how the application should ensure the integrity of the data it handles, and of its own operation and data.
- *Subsection 4.11, Availability:* Defines how the application should ensure the availability of the data it handles, and of its own resources and operation.
- *Subsection 4.12, Accountability:* Defines how the application should ensure the accountability of its users.

- *Subsection 4.13, Non-Repudiation:* Defines how the application should ensure non-repudiation of actions performed by its users.

- *Subsection 4.14, Preparation for Deployment:* Defines how to correctly prepare the application for installation and deployment.

Each of the requirements in these subsections includes the following information associated with the requirement:

- *Requirement:* A number assigned to the requirement, for cross-referencing purposes, and the brief text description of the requirement

- *Description:* More extensive, detailed information about what the requirement entails.

- *Assumptions and Constraints:* Any limiting assumptions that must be true for the requirement to be relevant and valid. Not all requirements will have assumptions or constraints associated with them, in which case they should be considered valid for all applications.

- *Test Objective:* The verification(s) to perform on an application to ensure that the requirement has been met.

- *Vulnerability Addressed:* Any of the top application vulnerabilities (listed in Section 3.1.1) that are addressed by the requirement. Not all requirements directly address any of these vulnerabilities. In some cases, requirements may only indirectly address one of the listed vulnerabilities. For requirements that only indirectly address a vulnerability, the vulnerability's number will be enclosed in parentheses in this column. For requirements that are driven by important factors (e.g., policy, best practices) but which do not address one of the listed vulnerabilities (directly or indirectly), there will be no cross reference in this column.

- *Policy Source:* Any policy, directive, manual, memorandum, or other official document(s) from which the requirement was derived. The numbers provided in this column are cross-references to the numbers assigned to the policy documents listed in Appendix B, "List of References". In the cases of requirements that are derived from best practices rather than documented policy, the entry in this column will indicate "BP" ("Best Practice").

- *Note:* Any other significant information related to the requirement. The "Note" column also includes cross-references to the comparable requirement(s) in *Recommended Standard Application Security Requirements* Version 1.1, to assist users in comparing the previous and current versions of this document. This cross reference is preceded by the text "V1.1:" followed by the relevant requirement number(s) from Section 4 of Version 1.1 of this document. Cross-references in italics point to Version 1.1 requirements that are partially addressed by the current requirement. Requirements in the current version for which there are no Version 1.1 cross-references should be considered new requirements *vis à vis* Version 1.1.

4.1 Assistance for Implementing these Requirements

The Application Security Developer's Guide contains extensive technical guidance on how to implement applications to satisfy all of the requirements in this document. In Section 2 of the Developer's Guide is a lookup table in which each requirement listed in Recommended Application Security Requirements is cross referenced to the Developer's Guide paragraph(s) addressing that requirement.

4.2 Exclusions from this Document

With the exception of some requirements pertaining to the use of HTML in Web applications, this document does not address requirements that pertain only to a specific programming language. For guidance on the secure implementation of applications in specific programming languages, please refer to the Application Security Developer's Guide.

4.3 Application Interaction with Underlying Host

This subsection lists requirements governing how the application interacts with its underlying host environment. The host environment, in this context, comprises the hardware platform and operating system, plus any "application environment" software, such as a database management system (DBMS) or Web server. These requirements apply to all applications, regardless of what security functions they perform.

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.1.1: No bypass of security controls	The application must prevent users from bypassing any application security controls in an attempt to directly access any underlying operating system, subsystem, or middleware component (i.e., PKI component, DBMS, Web server, etc.).		Verify that users cannot bypass application security controls to directly access underlying system components.	V3	BP	V1.1: 4.0.1
4.1.2: Integrity of host security data	The application must not perform, and must not be able to be used to perform, any function that may change the security configuration, security files, or security programs of the operating environment or platform in which the application runs.		Verify that the application does not modify, or enable modification of, security of its operating environment or platform.	V2, V3	BP	V1.1: 4.0.2
4.1.3: Integrity of system resources	(1) The application must not undermine or substitute the functionality or purpose of any files or programs —including security files and programs — belonging to an underlying operating system, subsystem or middleware component (i.e., PKI component, DBMS, Web server, etc.). (2) The application must not modify any files, programs or data—including security files, programs, or data— belonging to an underlying operating system, subsystem or middleware component (i.e., PKI component, DBMS, Web server, etc.).		(1) Verify that the application does not undermine or substitute operation of underlying system components' files or programs. (2) Verify that the application does not modify any underlying system components' security files, programs, or data.	V2	BP	V1.1: 4.0.3
4.1.4: Integrity of other applications' resources	The application must not modify in any unauthorized way any files, programs, or data belonging to other applications.		Verify that the application does not perform any unauthorized modifications of files, programs, or data belonging to other applications or underlying system components.	V2	BP	V1.1: 4.0.5
4.1.5: No compromising RPCs	(1) The application's remote procedure calls (RPCs) to a database or to another program (including application, DBMS, and system-level programs) must not modify in any unauthorized way any of the data, files, or programs belonging to the remote database or program. (2) The application's RPCs must not be used to modify files, programs, or data belonging to other applications or to any underlying system components (i.e., operating system, DBMS, Web server), or to achieve any other unauthorized action.		(1) Verify that the application that is using an external procedure call to access another program or a database remotely does not modify any security files, programs, or data associated with that program or database in any unauthorized way. (2) Verify that the application's remote procedure calls do not cause any unauthorized modification of files, programs, or data belonging to other applications or underlying system components.	V2	BP	V1.1: 4.0.4, 5.0.1, 5.0.2
4.1.6: Independent self-protection	The application design and implementation must not depend entirely on the presence of host or infrastructure security mechanisms to ensure that the application and its data are protected from compromise or denial of service.		Verify that the application contains, at a minimum, the error/exception handling functionality it requires to ensure it cannot be successfully targeted in a denial of service attack or compromise attack against its executables and the files it uses.	V3	29 (3.2.15.1)	

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.1.7: Operating environment verification	The application must be able to verify that its operating environment is properly configured and must report any deficiencies in that environment. This verification should include application identification of all conditions and dependencies the application needs to securely perform its functions. Specifically, the application must identify its dependencies on: (1) its host computer system (e.g., processor, primary and secondary memory capacity, 2) the underlying operating system (e.g. Version and release numbers, 3) subsystems (e.g., cryptography toolkits, 4) peripherals (e.g., network connection and speed, card readers, hardware tokens, 5) middleware (e.g., DBMS, Web server).		Verify that the application can recognize the correct operating state of its host environment and subsystems on which it relies, and that it does not proceed with execution if it detects that the host operating state is incorrect.	(V23)	16 (4.4)	
4.1.8: Secure configuration	(1) The application must be capable of being configured—preferably through automatic configuration—for secure operation in its intended environment(s), and must report any deficiencies that preclude its complete configuration. (2) The application's underlying host—including operating system, subsystems, and middleware—must be configured in compliance with all relevant STIGs. In addition, if there is a relevant STIG for the application itself, the application must be configured in compliance with that STIG.		Verify that the application includes an automatic installation/configuration utility that configures the application in a way that the default configuration options selected by the configuration utility are always the most restrictive (in terms of security) possible. Run COPS and other DISA FSO designated assessment tools and scripts to ensure that the application and its host are configured according to the relevant STIGs.	V29	2a (ECSC-1), 2b (ECSC-1), 2c (ECSC-1); 16 (4.4)	If an automated configuration process cannot be implemented in the application, easy manual configuration procedures must be documented, and administrators and users must be trained in how to perform these procedures.
4.1.9: Detection of external failures	The application must be able to detect failure conditions in the underlying host and surrounding infrastructure components with which it interfaces.		Verify that the application, throughout its execution, continues to detect the state of its host environment, and that it shuts down in an orderly fashion when it detects a failure in that environment. Verify also that the application, if it detects a failure, in an external infrastructure component (e.g., a cryptographic component), terminates its attempt to access that component, and terminates the function it was performing that required use of that component.	V23, (V15)	29 (3.2.4.1)	

4.4 General Use of Cryptography

This subsection lists requirements pertaining to the application's use of cryptography (including but not limited to public key cryptography), regardless of the purpose to which that cryptography is being put. This section does not include requirements that pertain only to a specific usage of cryptography, such as application invocation of a PKI for identification and authentication (I&A) of users, or application use of encryption to achieve confidentiality of data. Usage-specific cryptography-related application requirements are included in the following sections that pertain to security service implementations.

An application may invoke cryptographic functions via trustworthy calls to an external cryptographic mechanism—for example, a PKI or an encrypting file system—only if that external mechanism uses an approved cryptographic technology appropriate to the application function for which cryptography is being used (e.g., NSA Type 1, 3-DES, or AES implementations for Confidentiality; DOD PKI for I&A and Authorization).

Refer to section 4.3.3 of Department of Defense (DOD) Class 3 Public Key Infrastructure (PKI) Public Key-Enabled Application Requirements (13 July 2000) for a summary of the algorithms approved for particular cryptographic functions in FIPS 140-1 and in DOD PKI implementations.

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.2.1: Interoperability with DOD PKI	(1) Application PKI functions must be implemented by a PKI technology that is interoperable with the DOD PKI. (2) COTS application components that use or require PKI components must implement a PKI technology that is interoperable with the DOD PKI.		(1) Verify that the application has been PK-enabled using a PKI technology that is interoperable with DOD PKI. (2) Verify that the COTS component has received a certification of its interoperability with the DOD PKI from NSA, DTIC or another appropriate organization.	V3	4 (4.3-4.4); 29 (3.2.20)	V1.1: 4.0.6, 5.0.3
4.2.2: Class 4 certificates	The application must accommodate the transition from DOD PKI Class 3 certificates to Class 4 certificates with minimal modification of application code.	Application has already been PK-enabled, or is a legacy Mission Assurance Category 1 application in an unclassified environment and needs/uses DOD PKI cryptography	Verify that the application will accommodate use of Class 4 certificates with minimal modification of application code.	(V1, V23, V24)	3 (DOD PKI Certificate Assurance Levels; Evolution of DOD Certificates)	V1.1: 4.0.7
4.2.3: Applications to be PKE'd	The following application types must be PK-enabled by the deadlines specified in DOD PKE Policy to provide (at a minimum, unless otherwise noted) I&A, encryption, and digital signature functions that interoperate with the DOD PKI: (1) Unclassified private Web servers, including those that provide non-sensitive public releasable information; (1) All email applications; (2) Mission Assurance Category 1 applications operating on unclassified networks (exception: DMS High Grade Systems until DOD PKI is capable of satisfying High Grade System requirements); (3) Web applications (clients and servers) that run on unclassified networks; (4) Web client/browser applications on classified networks (for I&A to private Web servers); (5) All other types of applications that run on unclassified private DOD networks unless the unclassified network's predominant user community is not required to use DOD PKI certificates for authentication. Such users include retirees, dependents, and academia; (6) All email, Web, and legacy applications (client and server) in all operating environments by 30 September 2007	Application requires use of DOD PKI cryptographic capabilities	Verify that the application is PK-enabled to provide I&A, encryption, and digital signature functions interoperable with the DOD PKI. If not, verify that the exception has been noted and documented.	(V1, V5, V23)	3 (Web Server Access Control via PKI, Enabling of Networks and Applications); 4 (4.2, 4.5, 4.6, 4.7, 4.8); 29 (3.2.21)	V1.1: 4.0.8, 4.7.5 Applications that do not require use of DOD PKI cryptography do not need to be PK-enabled. However, an economic cost-benefit analysis comparing the costs of PKE of the application with whatever non-PKI alternative is being considered for providing the same security functions in the application. As per DOD PKE policy (4.2.1), Web server applications that are unclassified and publicly accessible, and contain information to which access must be limited only in order to (1) preserve copyright protections, or (2) facilitate its own development, or (3) make access to certain links available only to certain sites, are exempt from this requirement. (COMMENT: This note is not clear.)

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.2.4: Approval of cryptography	Cryptographic modules and other cryptographic technology (including key lengths, algorithm, certificate class, and token type) used by the application in connection with I&A, access control, confidentiality, integrity, or non-repudiation must be: (1) Certified by NIST as compliant with Federal Information Processing Standard (FIPS) 140 Level 1; (2) Approved by NSA if the Application is a MAC I application, or by NSA or NIST if the application is a MAC II application.		Verify that the application uses appropriately approved (NIST FIPS 140-1 certified or NSA-certified) cryptographic technology.	(V5, V24)	2a (DCNR-1), 2b (DCNR-1), 2c (DCNR-1); 16 (4.2.2); 29 (3.2.19.2)	V1.1: 4.0.9
4.2.5: SSL 3.0 and TLS 3.1	Web servers and browsers used in DOD applications should be fully compliant with the SSL 3.0 and TLS (SSL 3.1) specifications.	Application is a Web application	Verify that any web servers and browsers in use are fully compliant/compatible with the SSL 3.0 and TLS specifications.	V1, V5	29 (3.2.17.8)	
4.2.6: Client support for tokens	The application must support the DOD Common Access Card (CAC) and/or FORTEZZA card, as appropriate for the PKI used by the application.	Application is a client application (e.g., browser, user agent) that has been PK-enabled. No waiver has been granted allowing use of software certificates.	Verify that the application supports the use of DOD CAC/ FORTEZZA cards. If not, verify that an exception waiver has been granted for the use of software certificates.	(V1, V23)	3 (Enabling of Networks and Applications)	V1.1: 4.1.14
4.2.7: Certificate validation and revocation	The application must ensure that certificate validation and certificate revocation are performed correctly, and must not continue to use or accept invalid or revoked certificates.	Application invokes a PKI.	Verify that the invoked PKI responds correctly to CRLs, and does not honor revoked certificates.	V23	2d (ECRC-1), 2e (ECRC-1); 4 (4.4); 16 (4.3.2.4)	V1.1: 4.4.21
4.2.8: Cryptokey recovery	The encryption facility invoked by the application must perform the key recovery processes required by the cryptographic implementation.	Application uses cryptography	Verify that the invoked encryption facility operates in accordance with cryptokey recovery requirements.	V23	4 (4.4); 16 (4.3.1.4)	V1.1: 4.5.10
4.2.9: Key management, MAC I and MAC II	The application must use an NSA-approved cryptographic module to ensure that all key management functions associated with the application's use of cryptography are performed correctly. For asymmetric key management, the cryptographic module must use DOD PKI Class 3 or 4 Certificates stored on a hardware security token to distribute, store, and control generated keys.	Application uses cryptography. Application is MAC I or MAC II.	Verify that the key management functionality relied on by the MAC I or MAC II application has been implemented using NSA-approved cryptography.	V23	2a (IAKM-2), 2b (IAKM-2), 2d (IAKM-3); 16 (4.3.1)	Key management functions include: (1) Generating asymmetric key pairs; (2) Storing each key pair and related certificates; (3) Protecting stored private keys from compromise or loss; (4) Storing and protecting certificates that are trust points; (5) Escrowing or copying keys used for encryption for data recovery; (6) Importing and exporting key pairs and possibly related certificates.

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.2.10: Key management, MAC III	The application must use a NIST FIPS 140-1 evaluated cryptographic module to ensure that all key management functions associated with the application's use of cryptography are performed correctly. For asymmetric key management, the cryptographic module must use DOD PKI Class 3 certificates or pre-placed keying material to distribute, store, and control generated keys.	Application uses cryptography. Application is MAC III	Verify that the key management functionality relied on by the application has been implemented using FIPS 140-1 evaluated cryptography.	V23	2c (IAKM-1); 16 (4.3.1)	Key management functions include: (1) Generating asymmetric key pairs; (2) Storing each key pair and related certificates; (3) Protecting stored private keys from compromise or loss; (4) Storing and protecting certificates that are trust points; (5) Escrowing or copying keys used for encryption for data recovery; (6) Importing and exporting key pairs and possibly related certificates.
4.2.11: Interface to DOD PKI	The application must use Lightweight Directory Access Protocol (LDAP), Hypertext Transmission Protocol, or Hypertext Transmission Protocol over SSL (HTTPS) when communicating with the DOD PKI.	Application uses DOD PKI	Verify that the interface used by the application to access DOD PKI components is LDAP, and that all accesses of DOD PKI components by the application are made via SSL-encrypted connections.	V23	BP	To determine which interface protocol to use, see Department of Defense Class 3 Public Key Infrastructure Interface Specification (Draft, 13 January 2000).
4.2.12: Design supports PKI use	The application's design should not preclude it being configured or modified to use DOD PKI, and it should be able to use DOD PKI with only minor changes to its configuration or code.		Verify that the application has not been hard-coded in some way that will preclude it being configured or modified with minimal effort to interoperate with the DOD PKI.	V23	16 (4.4); 3 (Enabling of Networks and Applications)	V1.1: 4.0.6, *5.0.3*
4.2.13: DOD PKI certificates only	The application must be configurable to use only DOD PKI certificates, and to prevent use of any other type of certificate.	Application uses DOD PKI	Verify that the certificate validation capability relied upon by the application is able to distinguish between DOD PKI certificates and certificates originating elsewhere, and that it rejects any non-DOD PKI certificates.	V23	16 (4.4)	

4.5 Design and Coding

This subsection lists requirements regarding the correct design and coding of secure application software. These requirements apply to all applications (within the constraints defined in the Assumptions and Constraints for a given requirement), regardless of what security functions the application performs. Note that many of the following requirements are best practices for secure programming that are intended to ensure that the application program will operate as expected.

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.3.1: High-risk services	Avoid use of high-risk services and technologies, such as Telnet, Simple Network Mail Protocol (SNMP), mobile code, etc; in/by applications unless absolutely necessary.		Verify that the application uses no high-risk services, or that if the application does use a high-risk service, verify that a countermeasure, such as a security wrapper, has been implemented for the service to minimize the potential threat it poses.	V31	BP	V1.1: 4.0.10

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.3.2: One entry point and exit point	Every application process, including all security processes, should have only one entry point and one exit point.		Verify proper entry and exit points for each application process during the application development planning stage and maintain throughout application development cycle. After application development, verify application entry and exit points properly implemented and functional by conducting a 3rd party source code review.	V3, V28	BP	
4.3.3: Only invoked functions	The application should not include any functions that are not expressly invoked during application operation.		Verify that functions not explicitly invoked during application operation were removed/excluded during the application development cycle. After application development, verify functions not explicitly invoked during application operation were removed/excluded by conducting a 3rd party source code review.	V25	BP	
4.3.4: Only called runtime objects	The application runtime environment should exclude any software libraries, routines, or other resources that are not explicitly called by the application.		Verify that any runtime objects not explicitly called by the application were excluded/ removed during the application development cycle. After application development, verify runtime objects not explicitly called by the application were removed by conducting a 3rd party source code review.	V25	BP	
4.3.5: Least privilege	Each application process should have only the absolute minimum of privileges assigned to it that it requires to access the data or call the processes it needs.		Verify that least privileges were assigned to each application process during the application development planning stage and development cycle. After application development, verify least privilege assignment was properly implemented and functional by conducting 3rd party source code review.	V26	29 (3.2.15.3)	
4.3.6: Single tasking on single processors	Applications that run on single-processor hosts should be single-tasking, i.e., should execute only one task at a time, and should not initiate a new task until the previous task has completed execution.		Verify that the application is single tasked when used on single processor hosts. Verify that this requirement was designed into the application development planning stage and maintained throughout application development cycle. After application development, verify single tasking on single processor hosts properly implemented and functional by conducting 3rd party source code review.	V27	BP	
4.3.7: Avoid multitasking conflicts	In applications for multiprocessor hosts, multitasking or multithreading, must not create conflicts in usage of system resources (e.g., memory or disk addresses); all tasks and threads should be synchronized to prevent such conflicts.		Verify that multitasking conflicts are avoided on multiprocessor hosts. Verify that this requirement was designed into the application development planning stage and maintained throughout application development cycle. After application development, verify that the requirement was properly implemented and functional by conducting 3rd party source code review.	V27	BP	
4.3.8: Small, simple modules	The application should consist of multiple small, simple, single-function modules instead of one large, complex module that performs multiple functions.		Verify that small, simple, single function modules were designed intothe application during its development planning stage and development cycle. After application development, verify that small, simple, single function modules were properly implemented and functional by conducting 3rd party source code review.	V28	BP	

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.3.9: Self-contained modules	Each application module should be designed to be self-contained and atomic, so any module can be disabled when not needed or if found to be vulnerable or errored without affecting the operation of any other modules.		Verify that self-contained modules were designed into the application during its development planning stage and development cycle. After application development, verify that self-contained modules were properly implemented and functional by conducting 3rd party source code review.	V3	BP	
4.3.10: Minimal trusted modules	The application should contain very few trusted modules; the only modules that are trusted should perform critical security control-related operations, such as I&A, auditing, access control, etc.		Verify that minimal-trusted modules and trusted modules only perform critical security control related operations. Verify that this requirement was designed into the application development planning stage and maintained throughout application development cycle. After application development, verify minimal trusted modules with trusted modules that only perform critical security control related operations were properly implemented and functional by conducting 3rd party source code review.	V26	BP	
4.3.11: Untrusted module limitations	Untrusted application modules should be unable to access security data, functions, or privileges.		Verify that untrusted modules are designed with appropriate security limitations during the application development planning cycle. After application development, verify untrusted module limitations by conducting 3rd party source code review.	V2	BP	
4.3.12: No user interface bypass	The application should be designed to prevent users from bypassing any user interface software to directly access application data or processes.		Verify that users are unable to bypass the application interface. Verify that this requirement was designed into the application development planning stage and maintained throughout application development cycle. After application development, verify prevention of user interface bypass was properly implemented and functional by conducting 3rd party source code review.	V3	BP	
4.3.13: Separate data and programs	The application should write any files it creates to a different directory from that in which the application executable code is stored.		Verify that the application was designed to separate data and program directories. Verify that this requirement was designed into the application development planning stage and maintained throughout application development cycle. After application development, verify separate data and program directories are properly implemented and functional by testing and conducting source code review.	V29	2a (DCPA-1), 2b (DCPA-1); BP	
4.3.14: NIAP-certified components	All third-party (COTS, GOTS, or Open Source) components used in the application to perform security functions must be NIAP-certified at the level of assurance and robustness that is appropriate for the classification of the data handled by the application (and of the network on which that application will transmit the data), unless a written waiver is granted allowing their use without NIAP certification (or allowing an alternative certification).	No waiver has been granted waiving this requirement.	Verify that all third-party components used in the application to perform security functions appear on the NSA's Common Criteria Evaluated Products List found at: http://www.radium.ncsc.mil/tpep/epl/cc_st.html.	V30	1 (4.17, 4.19); 2a (DCPD-1), 2b (DCPD-1), 2c (DCPD-1), 2d (DCAS-1), 2e (DCAS-1), 2f (DCAS-1)	
4.3.15: No collection of user-identifying data	The application must not use cookies or any other Web technology that collects user-identifying information such as extensive lists of previously visited sites, email addresses, or other information to identify or build profiles on individual users.	Application is a Web server application used for a publicly-accessible Website	Verify that the web server application does not collect any user-identifying data.	V15, V19, (V2, V5)	BP	

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.3.16: Avoid persistent cookies	The application must not use persistent cookies to collect non-identifying information about users unless the following is true: (1) The Website issues a clear and conspicuous notice that cookies are being used to collect data about users, the reason why they are being used, and the type of information that is being collected. (2) There is a compelling need for the site to gather such data. (3) The Web server has implemented appropriate and publicly disclosed privacy safeguards for handling information derived from cookies. (4) The Secretary of Defense has approved in writing the use of the data-collecting persistent cookies.	Application is a Web server application used for a publicly-accessible Website	Verify that, if using persistent cookies, the application: (1) Issues a clear and conspicuous notice that cookies are being used to collect data; (2) Has a compelling need for the site to gather such data; (3) Has implemented appropriate and publicly disclosed privacy safeguards for handling information derived from cookies; (4) Has been approved, in writing, by the Secretary of Defense to use data-collecting persistent cookies.	V15, V19, (V2, V5)	BP	
4.3.17: Invocation of trusted processes	Trusted application processes must not be allowed to be invoked by end-users or user-controlled processes.		Verify that the application properly invokes trusted processes. Verify that this requirement was designed into the application development planning stage and maintained throughout application development cycle. After application development, verify invocation of trusted processes properly implemented and functional by conducting 3rd party source code review.	V3, (V31)	BP	
4.3.18: No security through obscurity	DOD applications should not rely on "security through obscurity". Java applications should not be implemented using bytecode obfuscation.		Verify bytecode obfuscation is not part of the planned security implementation for a Java application. Verify that this requirement was designed into the application development planning stage and maintained throughout application development cycle. After application development, verify bytecode obfuscation was not implemented by conducting 3rd party source code review.	(V2, V28)	BP	
4.3.19: Coding for operational environment	Application code should be designed and written to operate under the constraints of the operational host (i.e., after STIG configuration) and infrastructure. Applications should not expect to use unavailable services, to be granted unauthorized permissions, or to run without operational security constraints.		Verify application code was written to meet the constraints of the operational host environment. Verify that waivers to use unavailable services or grant unauthorized permissions are not needed.	(V3, V23, V25, V26, V29, V31)	BP	
4.3.20: No trusted programs invoking untrusted programs	The application must contain no trusted programs that invoke untrusted programs.	Application contains third-party component(s)	Verify no invocation of untrusted programs by trusted programs. Verify that this requirement was designed into the application development planning stage and maintained throughout application development cycle. After application development, verify no invocation of untrusted programs by trusted programs through conduction of 3rd party source code review.	V3, (V26)	BP	
4.3.21: No GoTo statements	Application code should not include GoTo statements that obscure the control flows within the program		Verify that "GoTo" statements are not used within application code, during the application development cycle. After application development, verify that "GoTo" statements were not used by conducting 3rd party source code review.	V28	BP	

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.3.22: No unnecessary software	The application code base should not include any unnecessary software (custom-developed or third-party).		Verify that unnecessary software is not used by the application. Verify that this requirement was designed into the application development planning stage and maintained throughout application development cycle. After application development, verify unnecessary software was not used by conducting 3rd party source code review.	V25	BP	
4.3.23: Secure data types	Applications should use only secure data types (e.g., signed rather than unsigned values in C and C++).		Verify only secure data types are used within the application by conducting 3rd party source code review and scanning with the appropriate source code scanning tools.	V31	BP	
4.3.24: Safe system calls	Applications, regardless of the programming language in which they are implemented, should use only safe system calls, i.e., calls that do not make the application vulnerable to buffer overflows or other types of attacks.		Verify that only safe system calls were designed to be used within the application by conducting 3rd party source code review and scanning with the appropriate source code scanning tools.	V31	BP	
4.3.25: Appropriate APIs	Applications should use only Application Programmatic Interfaces (APIs) intended for use by software processes, and not interfaces intended for use by human users.		Verify that proper usage of APIs was designed into the application development cycle. After application development, verify API usage properly implemented and functional by conducting a 3rd party source code review.	V3	BP	
4.3.26: No escapes to shell or command line	Applications should not include command-line or shell escape codes.		Verify that the application contains no command-line or shell escape codes. Verify that this requirement was designed into the application development planning stage and maintained throughout application development cycle. After application development, verify no command-line or shell escape codes by conducting 3rd party source code review and scanning with the appropriate source code scanning tools.	V31	BP	
4.3.27: Avoid escape codes	Application code should not include escape codes that invoke system level or device level functions.		Verify that the application contains no device escape codes. Verify that this requirement was designed into the application development planning stage and maintained throughout application development cycle. After application development, verify no device escape codes by conducting 3rd party source code review and scanning with the appropriate source code scanning tools.	V31	BP	Escape codes in applications can inadvertently or intentionally cause denial of service (e.g., to a device) or bypass of system security controls. Examples of escape codes to be avoided include but are not limited to: (1) escape codes that invoke the system shell or command line; (2) escape codes in the Hayes modem command set; (3) VT100 escape codes.
4.3.28: Current patches	Applications that include third-party code should include the absolute latest versions of that code, with all security patches applied, in accordance with the IAVA Implementation Process.		Verify that all 3rd party code is the latest patched version. Verify that this requirement was met during the application development cycle. After application development, verify all 3rd party code is the latest patched version by conducting a 3rd party source code review.	V30	BP	

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.3.29: Consistent naming	Aliases, pointers, links, caches, and other objects named in the application should be named consistently throughout the program. The application should use symbolic naming and dynamic linking, with globally unique names and symmetrical treatment of aliases.		Verify that naming conventions are consistently implemented throughout application development cycle. After application development, verify consistent naming was properly implemented within the application by conducting a 3rd party source code review.	V28	BP	
4.3.30: Frequent cache clearing	Applications should be clear their own caches frequently.		Verify that applications are designed to frequently clear any cache used. Verify that this requirement was met during the application development cycle. After application development, verify frequent cache clearing is properly implemented and functional by conducting a 3rd party source code review.	(V2)	BP	
4.3.31: Asynchronous consistency	Applications should avoid timing and sequence errors (including race conditions, incorrect synchronization, and deadlocks). This should be achieved by coding all transactions to be atomic, implementing multiphase commits, and using hierarchical locking strategies.		Verify that the application was designed to avoid timing and sequence errors during the application development cycle. After application development, verify asynchronous consistency is properly implemented and functional by conducting a 3rd party source code review.	V27	BP	
4.3.32: No off-by-one counting errors	Applications should not include off-by-one counting errors.		Verify that the application does not include any off-by-one counting errors by conducting a 3rd party source code review.	(V15)	BP	
4.3.33: No fixed buffer sizes	When defining buffer sizes, do not define an absolute value or constant to fix the buffer size. Instead, use relative values or do not specify buffer size constraints at all.		Verify that the application does not rely on any absolute values or fixed buffer sizes by conducting a 3rd party source code review.	V7	BP	
4.3.34: Do not omit negations	Applications should not omit necessary negations.		Verify that the application does not omit any necessary negations by conducting a 3rd party source code review.	(V15)	BP	
4.3.35: Assign minimal resources	Each application process should have the absolute minimum computer resources made available to it that it needs to operate.		Verify that the application was designed to have the absolute minimum of computer resources made available to each application process. Verify that this requirement was met during the application development cycle. After application development, verify minimal computer resources to application processes was properly implemented and functional by conducting a 3rd party source code review.	V27	BP	
4.3.36: Results buffer larger than source	Buffers used by the application to store results must always be larger than the source buffers from which the data that will fill the results buffer originate. (COMMENT: This might use a little more explanation.)		Verify that the application was designed so that its results buffers are always larger than source buffers. Verify that this requirement was met during the application development cycle. After application development, verify the result buffer is always larger than the source buffers by conducting a 3rd party source code review in addition to scanning with the appropriate source code scanning tools.	V7	BP	

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.3.37: No passing of long data elements	Application library functions must not pass excessively long data elements into other libraries. Application functions should reject excessively long data elements passed to it by third-party library functions used in the application.		Verify that the application does not pass long data elements between libraries. Verify that data elements defined for the application are as terse as possible. Verify that this requirement was met during the application development cycle. After application development, verify long data elements are not passed between libraries by conducting a 3rd party source code review.	V7	BP	Third-party software library routines cannot be trusted not to cause internal buffer overflows, even when the custom-developed code in the application does not cause them. It is good programming practice to avoid coding custom-developed library functions that pass long data elements. Indeed, the best practice is to define all data elements handled by the application to be as terse as possible.
4.3.38: Do not trust operating system variables	The application must not trust operating system environment variables. Instead, the program should pass every argument to the application in an environment parameter.		Verify that the application does not trust operating system variables. Verify that this requirement was met during the application development cycle. After application development, verify that operating system variables are not trusted by conducting a 3rd party source code review.	V29, (V8)	BP	
4.3.39 Read only database front-ends	The database front-end functionality should allow only query (read) access, and no update (write) or delete access.	Application is a Web-based front-end to a database	Verify that database front-ends are designed to be read only. Verify that this requirement was met during the application development cycle. After application development, verify database front-end read only query was properly implemented and functional by conducting a 3rd party source code review.	V2	BP	
4.3.40: Discrete database transactions	Database updates should be implemented as discrete transactions.	Application updates a database:	Verify that the application uses discrete database transactions. Verify that this requirement was met during the application development cycle. After application development, verify requirement by conducting a 3rd party source code review.	V28	BP	
4.3.41: CPU time allotment	The application should not prevent the developer (or administrator) from configuring a maximum limit on the amount of CPU usage time allotted to any single application process, and the application must not grant additional CPU time to any process that reaches that limit.		Verify that the application does not interfere with the configuration of maximum CPU usage time allotments to individual application processes. Verify that the application does not override any configured CPU usage time allotments and grant additional CPU usage time to processes that have reached their configured maximum usage time.	V27	BP	

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.3.42: Category 1 and Category 2 mobile code signing	The application must digitally sign the mobile code before releasing it using an NSA - or NIST-approved PKI code signing certificate to sign the code.	(1) Application is Web server or other server application that acts as the source for Category 1 or Category 2 mobile code. (2) Responsible CIO has not signed written waiver allowing use of commercial certificates for signing Category 1 and Category 2 mobile code. (3) Responsible CIO has not signed written waiver allowing dissemination of unsigned Category 1 or Category 2 mobile code.	Verify that the application digitally signs mobile code before release using appropriately approved PKI code signing certificate. If this is not the case, verify that the application that uses commercial certificate to sign mobile code has received a written waiver from the responsible CIO allowing use of commercial certificates. If the application is still not compliant with the requirement, verify that the application that does not sign Category 2 mobile code has received a written waiver from the responsible CIO allowing mobile code with no digital signature.	V30, V32, (V2)	2a (DCMC-1), 2b (DCMC-1), 2c (DCMC)-1; 5 (1.1.3, 1.2.4)	V1.1: 4.8.1 The appropriate approver (NIST or NSA) for the code signing certificate will be determined by the Mission Assurance Category of the application.
4.3.43: Signed Category 1 and Category 2 mobile code execution	Before executing signed Category 1 or Category 2 mobile code, the application must validate the digital signature on the mobile code to ensure that the code originated from a trusted source.	Application is Web or other client application. Mobile code has been received from a trusted source as proved by signature on the code	Verify that the application validates digital signature on mobile code before executing it.	V30, V32, (V2)	2a (DCMC-1), 2b (DCMC-1), 2c (DCMC)-1; 5 (1.1.3); 16 (4.3.3.1)	V1.1: 4.8.2
4.3.44: Unsigned Category 2 mobile code execution	The application shall not execute unsigned Category 2 mobile code unless it is implemented in a way that ensures that it executes in a constrained environment (i.e., a "sandbox") without access to local OS and network resources (e.g., file system, Windows registry, and network connections other than to its originating host).	Application is Web or other client application. Mobile code has been received from an untrusted source and is not signed	Verify that Category 2 mobile code executes with no access to local OS and network resources (except for network connections to code's originating host).	V2, V30, V32	2a (DCMC-1), 2b (DCMC-1), 2c (DCMC)-1; 5 (1.2.3-4)	V1.1: 4.8.3
4.3.45: Category 2 mobile code notification	The application must be configurable to warn users when the application is about to execute Category 2 mobile code.	Application is Web or other client application	Verify that the application can be configured to issue warning to user when application is about to execute Category 2 mobile code.	V30, V32, (V2)	2a (DCMC-1), 2b (DCMC-1), 2c (DCMC)-1; 5 (1.2.4)	V1.1: 4.8.4
4.3.46: Category 3 mobile code handling	The application may act as a source for, or may execute, Category 3 mobile code after a risk assessment has been performed, and appropriate risk mitigation has been undertaken.		Verify that appropriate risk management is being performed for application that uses Category 3 mobile code.	V30, V32	2a (DCMC-1), 2b (DCMC-1), 2c (DCMC)-1; 5 (1.3.4)	V1.1: 4.8.5
4.3.47: No "emerging" mobile code	The application must not act as a source for, or execute, emerging mobile code technology.	Written waiver has not been granted to CIO (as per [5 (1.4.3)]) allowing use of emerging mobile code technology	Verify that the application's "emerging mobile code technology" has received the necessary waiver.	V30, V32, (V2)	2a (DCMC-1), 2b (DCMC-1), 2c (DCMC)-1; 5 (1.4.2)	V1.1: 4.8.6
4.3.48: Mobile code in email	The application must disable/prevent (or interface in trustworthy way with another program that can disable) the execution of mobile code in message bodies or attachments.		Verify that email client disables/prevents execution of mobile code in message bodies and attachments.	V9, V32	2a (DCMC-1), 2b (DCMC-1), 2c (DCMC)-1; 5 (2.1); 27 (6.6.4.5)	V1.1: 4.8.7
4.3.49: Email client mobile code notification	The application must be configurable to issue a warning to the user, before opening an email attachment, that the attachment about to be opened may contain mobile code.		Verify that the email client can be configured to notify user, before opening email attachment, that attachment may contain mobile code.	V30, V32	2a (DCMC-1), 2b (DCMC-1), 2c (DCMC)-1; 5 (2.2)	V1.1: 4.8.8

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.3.50: Sandboxing of mobile code	The application must provide a "sandbox" mechanism and ensure that the mobile code can execute only within that sandbox.		Verify that the application is designed to sandbox mobile code. Verify that this requirement was met during the application development cycle. After application development, verify proper implementation and functionality of sandboxing by conducting a 3rd party source code review.	V2, V30, V32	29 (3.2.5.13)	
4.3.51: Disabling of unsigned mobile code	The application must be able to disable operation of unsigned Category 1 mobile code.		Verify an application's exemption from meeting mobile code requirements by determining if the application utilizes the types of mobile code listed on the left.	V2, V30, V32	2a (DCMC-1), 2b (DCMC-1), 2c (DCMC)-1; 5 (1.1.4)	
4.3.52: Unrestricted mobile code types	The application need not satisfy any of the above mobile code-related requirements for the following types of mobile code: (1) Scripts and applets embedded in or linked to Web pages and executed in the context of the Web server (e.g., Java servlets, Java Server Pages, Java RNI, Java Jini, CGI, Active Server Pages, Cold Fusion Markup Language (CFML), PHP Hypertext Processor (PHP), Server Side Include (SSI), server-side JavaScript, and server-side LotusScript). (2) Local programs and command scripts (e.g., binary executables, shell scripts, batch scripts, Windows Scripting Host [WSH], Perl scripts). (3) Distributed object-oriented programming systems (e.g., Common Objective Request Broker Architecture [CORBA], Distributed Component Object Model [DCOM]). (4) Software patches, updates, including self-extracting updates and software updates that must be invoked explicitly by the user (e.g., Netscape SmartUpdate, Microsoft Windows Update, and Netscape Web browser plug-ins).		Verify the use of any mobile code is of an allowed type of mobile code. After application development, verify the use of any mobile code meets the requirements for allowed types of mobile code by conducting a 3rd party review.	(V30, V32)	2a (DCMC-1), 2b (DCMC-1), 2c (DCMC)-1; 5 (Attachment 1)	
4.3.53: Allowable mobile code types	The application may contain or execute the following types of mobile code, but only in conformance with the mobile code-related requirements defined in References 4.3.42-4.3.51: (1) Category 1: ActiveX Windows Scripting Host, when used to execute mobile code; UNIX shell scripts, when used as mobile code; DOS batch scripts, when used as mobile code. (2) Category 2: Java applets and other Java mobile code; Visual Basic for Applications (VBA); LotusScript; PerfectScript; Postscript. (3) Category 3: Javascript (include Jscript and ECMAScript variants); VBScript; Portable Document Format (PDF); Shockwave/Flash. (4) Emerging Mobile Code: Any mobile code type not listed here.		Verify that the application digitally signs mobile code before release using appropriately approvedPKI code signing certificate. If this is not the case, verify that the application that uses commercial certificate to sign mobile code has received a written waiver from the responsible CIO allowing use of commercial certificates. If the application is still not compliant with the requirement, that the application that does not sign Category 2 mobile code has received a written waiver from the responsible CIO allowing mobile code with no digital signature.	V2, V30, V32	1 (4.24); 2a (DCMC-1), 2b (DCMC-1), 2c (DCMC)-1; 5 (Attachment 1)	

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.3.54: Strictly controlled remote administration	Application administration tools that support remote administration must provide at least the same levels of assurance, robustness, and protection of privileged functions and sensitive data as the the same tools when used on an administration console direct-connected to the server. Specifically, the levels of assurance, robustness, and protection must ensure that the confidentiality and integrity of administration data transmitted between the tool and the server application are at least the same if not greater than they would be were the administration console direct-connected to the server, and that the authentication, authorization, and accountability of the administrator using the tool is no less secure than it would be were the administrator using the tool locally, direct-connected to the server.	There is a compelling operational need for remoted administration. Application is a server application, and includes administration tools that can be used to administer the application remotely.	Verify that the remote administration tool uses appropriate security mechanisms such as cryptography (e.g., SSL with appropriate Class of X.509 certificates if the tool is browser-based) to guarantee that the administration session is at least as secure as it would be were the tool used locally, direct-connected to the administered server application.	V37	2d (EBRP-1), 2e (EBRP-1)	Remote performance of privileged functions in connection with administration of applications is strongly discouraged, permitted only for compelling operational needs, and must be strictly controlled.
4.3.55: Secure programming language	The programming languages in which the application is written must not have characteristics that could create vulnerabilities in the application unless effective countermeasures in the application's design and coding fully counteract those vulnerabilities.		Verify that the application does not include languages that are known to introduce vulnerabilities, such as untyped languages like PHP and VBScript, or buffer-overflow prone languages like C or C++. If the application does include such a language, verify that the application includes countermeasures that fully counteract the vulnerabilities created by the language, e.g., no fixed buffer sizes, effective input validation and bounds checking to counteract the potential for buffer overflow.	V39	BP	
4.3.56: Dispersion of security code	The application's security processes should not be dispersed throughout the application code, but should be located close together (ideally contiguously).		The application is written in multiple, likely non-interoperable, programming languages. This vulnerability must be avoided in the application's design. A code review after the application is built may help locate the application's security processes, but if those processes are dispersed throughout the code base, little if anything can be done to rectify the problem so late in the development lifecycle.	V38	BP	
4.3.57: No buffer overflows in library routines	The application must not call any software libraries that contain buffer overflow vulnerabilities.		Verify that the application only uses software libraries that have been tested for security vulnerabilities and approved for use.	V7	BP	This requirement is particularly important for library routines in C or C++.
4.3.58: Functional Architecture	The application's design includes a functional architecture that complies with the requirements in DODI 8500.2.		Verify that the design document for the application includes or is associated with a Functional Architecture Document, and that this document complies with the content requirements defined in DODI 8500.2.		2a (DCFA-1), 2b (DCFA-1), 2c (DCFA-1)	

4.6 Identification and Authentication (I&A)

This subsection lists the security requirements defining how server application processes should perform—or interface with external I&A facilities that perform on their behalf—identification and authentication of users, client processes, and other servers. An application may invoke an external I&A mechanism, such as a Single Sign-on system, via trustworthy calls, only if that external I&A mechanism is implemented by an approved technology. See IATF (Reference 27) Section 4.3.1 for a discussion of the processes and technologies involved in identification and authentication.

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.4.1: User authentication	Application must ensure that users have been authenticated before granting them access to sensitive resources or trusted roles.		Verify that application ensures that users have been authenticated before granting them access to sensitive resources or trusted roles.	V1	2d (IAIA-2), 2e (IAIA-2); 16 (4.2.3); 29 (3.2.1, 3.2.17.1)	V1.1: 4.1.1
4.4.2: Process authentication	Applications must authenticate any process, program, or other active entity or object that interacts with the application on a user's behalf.		Verify that this authentication requirement was met during the application development planning stage and maintained throughout application development cycle. Post application development cycle, verify that the requirement was properly implemented by testing.	V1	BP	
4.4.3: Authentication warning notification	Before granting an authenticated user access to the application's resources, the application must present a warning notification message to that user containing the following information: (1) user has accessed a government system; (2) extent to which the application will protect the user's privacy rights, (3) highest sensitivity level/classification of data that may be handled by the application, (4) user's actions are subject to audit, (5) user's responsibilities for handling sensitive or classified information when using the application.		Verify that upon authentication, application presents required information to the user before granting access to application resources.	(V2)	1 (4.23); 2d (ECWM-1), 2e (ECWM-1), 2f (ECWM-1); 29 (3.2.1.4.5, 3.2.8.1, 3.2.8.2)	V1.1: 4.1.29
4.4.4: Classified authentication warning notification	In addition to the information in 4.4.3 above, the application's warning notification message to authenticated users must include the following additional information associated with the authenticated UserID: (1) date, time, origination (e.g., client/browser IP address or domain name) of most recent previous login; (2) number of unsuccessful login attempts by the UserID since the last successful login.	The application handles classified information	Verify that upon authentication, application presents this information in addition to the information in 4.4.3 to the user before granting access to application resources.	(V2)		V1.1: 4.1.29
4.4.5: I&A mechanisms	(1) I&A performed before granting access to the application must use a non-forgeable, non-replayable mechanism that supports both one-way and two-way authentication. (2) Application I&A should use one or more of the following technologies instead of or in addition to UserID/static password: (a) single sign-on (SSO, 2) PKI, (b) hardware token (CAC or FORTEZZA, 4) biometrics, (c) dynamic (one-time) passwords.		(1) Verify that I&A is performed using a non-forgeable, non-replayable mechanism that supports one-way and two-way authentication. (2) Verify that the application's I&A process uses SSO, PKE, hardware token, and/or biometrics.	V1	2a (DCBP-1), 2b (DCBP-1), 2c (DCBP-1), 2d (IAIA-2), 2e (IAIA-2); 27 (4.5.3.5); 29 (3.2.1.4.4, 3.2.1.8, 3.2.17.1.1)	V1.1: 4.1.2, 4.1.3
4.4.6: Authentication chain of trust	For every user session and transaction, the application must ensure that an authentication chain of trust is established and maintained between the client/browser, the application server, and any backend servers used by the application.		Verify that the application establishes and maintains the necessary authentication chain of trust.	V1, V3, V18	BP	V1.1: 4.1.4

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.4.7: I&A trusted path	All I&A transactions must be performed over a trusted path (e.g., encrypted link) between the entity to be authenticated and the entity performing the authentication.		Verify that all I&A transactions conducted by the application are performed over a trusted path.	V1	29 (3.2.2)	V1.1: 4.1.5
4.4.8: Trusted path initiation	The trusted path used for user I&A must be initiated by the user, not by the application.		Verify that the application never attempts to initiate an I&A trusted path on its own.	V1	29 (3.2.2.2)	
4.4.9: I&A data at rest	The application must ensure that all passwords, encryption material, and any other sensitive data it uses in the I&A process are adequately protected from disclosure or tampering when at rest, and that the application itself, and the interfaces between the application and I&A data stored in the file system, RDBMS, Web server, etc; cannot be exploited to compromise those data.		Verify that the application ensures that all sensitive data are adequately protected at rest.	V2, V3	29 (3.2.1.5.2)	
4.4.10: Backend system I&A	The application must not prevent the backend system from authenticating users or interfere in any way with that authentication process.	The application is a server application that interfaces with a "backend" system or DBMS that requires users to be authenticated to it before allowing them access	Verify that server application through which users gain access to a backend system enables the backend system to authenticate users if required.	V1, V3	BP	V1.1: 4.1.6
4.4.11: I&A on behalf of backend system	The application should use only accredited authentication portal or single sign-on (SSO) technology to accomplish I&A on behalf of a "backend" system or DBMS.	The application is a server application that performs user I&A on behalf of a "backend" system or DBMS application.	Verify that server application performing I&A on behalf of a backend application does so using accredited authentication portal or SSO technology.	V1, V3	BP	
4.4.12: Unsuccessful I&A attempts	The I&A mechanism must enable administrator configuration of the maximum number of login attempts (configurable per user or per role) allowed within a given time period.		Verify that the I&A mechanism enables the administrator to configure the maximum number of unsuccessful login attempts allowed per user and per role.	V1	29 (3.2.1.6)	V1.1: 4.1.7
4.4.13: I&A lockout period	The application I&A mechanism must enable administrators to configure the duration of the "lockout" period during which a user (or role) who exceeds the number of allowable login attempts will be prevented from making another I&A attempt.		Verify that the I&A mechanism enables the administrator to configure the I&A lockout period for a user or role.	V1	2d (ECLO-2), 2e (ECLO-1); 29 (3.2.1.6.1)	V1.1: 4.1.8
4.4.14: No roles without authentication credential	The application must not allow any role to be defined without an associated authentication credential (e.g., password, if UserID/password I&A is used; or PKI certificate, if PKI-based I&A is used).	Application implements Role Based Access Control	Verify that the application does not allow a role to be defined without requiring it have a role-associated authentication credential assigned to it.	V1	BP	V1.1: 5.2.3
4.4.15: Group/Role I&A	The application must first individually authenticate every user who claims membership in a group or role before performing subsequent group/role level I&A for that user, and must also authenticate the group/role based on its group/role authenticator. This group/role authenticator must be a DOD PKI certificate unless a waiver has been granted by the responsible DAA.	Application performs I&A at the group or role level.	Verify that the application does not perform group or role I&A for a user without first authenticating individual user claiming membership in the group or role. Verify that the group/role authenticator is a DOD PKI certificate, or that a waiver has been granted by the responsible DAA allowing use of another type of authenticator.	V1	2d (IAGA-1), 2e (IAGA-1)	V1.1: 4.1.23

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.4.16: Interprocess authentication	Application processes that act on behalf of users should be authenticated by the server or peer process with which they are attempting to interoperate. The strength of interprocess authentications should be at least as great as the strength of user authentications performed by the same application.		Verify that any server or peer process interoperations are authenticated at a level equal to, or greater, the strength of user authentication performed by the application.	V1	2a (DCNR-1), 2b (DCNR-1), 2c (DCNR-1); 29 (3.2.1.7)	
4.4.17: Client-Server Interprocess I&A	Before accepting a request from any client process, the application's server process must authenticate that client process using an interprocess authentication technology approved by NSA or NIST and appropriate for the application's Mission Assurance Category (e.g., X.509/SSL or Kerberos).	The application is a distributed client-server application that: operates in a high level of concern environment in which the security protections are not adequately robust, and performs sensitive functions or handles high-value information.	Verify that before accepting requests from a client process, the server process authenticates that client process using appropriate, approved technology.	V1	2a (DCNR-1), 2b (DCNR-1), 2c (DCNR-1); 29 (3.2.1.7)	V1.1: 4.1.27
4.4.18: Peer-to-Peer Interprocess I&A	Before accepting any requests from one another, the application's peer processes must mutually authenticate one another using an interprocess authentication technology approved by NSA or NIST and appropriate for the application's Mission Assurance Category (e.g., X.509/SSL or Kerberos).	The application is a distributed client-server application that operates in a high level of concern environment in which the security protections are not adequately robust, and performs sensitive functions or handles high-value information	Verify that peer processes authenticating using appropriate, approved technology before accepting requests from one another.	V1	2a (DCNR-1), 2b (DCNR-1), 2c (DCNR-1); 29 (3.2.1.7)	V1.1: 4.1.28
4.4.19: Strong I&A of trusted users	The application must require strong authentication of users who perform administrative or other trusted functions before granting those users access to any of the application's administration or trusted processes.		Verify that administrative and other trusted users can only be authenticated by an application using strong authentication techniques. Anything less should fail.	V1	BP	
4.4.20: I&A using PKI certificates	The application must support the PKI (X.509) certificate class appropriate to the application's Mission Assurance Category: (1) Mission Assurance Categories I and II: DOD PKI Class 4 certificates on tokens, (2) MAC III: Class 3 certificates on tokens or used with SSL/TLS.	The application performs certificate-based I&A.	Verify that I&A mechanism uses the Mission Assurance Category-appropriate type of certificate.	V1	1 (4.8.2); 2a (IATS-2), 2b (IATS-2), 2c (IATS-1); 3 (Selection of appropriate DOD PKI certificate assurance levels); 29 (3.2.1.4.3)	V1.1: 4.1.9
4.4.21: Two-way mutual authentication	The application must perform two-way mutual authentication, i.e., client must authenticate server, based on the DOD PKI X.509 Server certificate; and server must authenticate client, based on the client's DOD PKI X.509 personal identity certificate.	The application is a Web application	Verify the two-way mutual authentication requirement was met during the application development planning stage and maintained throughout application development cycle. Post application development cycle, verify that the requirement was properly implemented and functional by testing the application and conducting 3rd party source code review.	V1	29 (3.2.1.7)	

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.4.22: Certificate validation	The Application must be programmed to interpret the directory-stored Certificate URL directly for certificate validation. If this is not the case, the application and the directory it uses must enable the application to validate the certificate in real time.	The application is PK-enabled server application. The local directory used by application can store Certificate URLs issued by DOD PKI CA.	Verify that the application directly interprets the Certificate URL. If not, verify that the application enables real time certificate validation.	V23	BP	
4.4.23: Session tokens	The application should use only the user's X.509 certificate or an encrypted one-time (non-persistent) cookie as the session reauthentication token. No other mechanism (i.e., passwords embedded in HTML form fields, URLs, or persistent or unencrypted cookies) may be used.	The application is a Web application that uses session tokens for user reauthentication	Verify the requirement to use only X.509 certificates or encrypted one-time cookies was met during the application development planning stage and maintained throughout application development cycle. Post application development cycle, verify that the requirement was properly implemented and functional by testing.	V1, (V5, V19)	BP	
4.4.24: I&A using PKI tokens	The application must support the type of NSA-certified token appropriate to the application's Mission Assurance Category: (1) MAC I and MAC II: FORTEZZA, common access card (CAC), or another NSA-approved Class 3 or Class 4 hardware token; MAC III: CAC or software token (until DOD PKI transition to CAC)	The application performs token-based I&A	Verify that client application I&A supports the Mission Assurance Category-appropriate tokens, and that these tokens have been NSA-certified.	V1	2a (IATS-2), 2b (IATS-2), 2c (IATS-1); 3 (PKI Tokens); 29 (3.2.1.4.2)	V1.1: 4.1.10
4.4.25: Support for CAC	The application must be able to accommodate use of the CAC by the ASD C3I-defined deadline, with minimal change to the application code.	Application performs I&A based on certificates stored in software tokens	Verify that I&A mechanism can accommodate use of CAC with little or no change to application code.	V1	3 (PKI Tokens)	V1.1: 4.1.17
4.4.26: Browser support for tokens	By the ASD C3I-defined deadline for migration to hardware tokens, browsers, including those that already support software tokens, must provide the necessary APIs and plug-ins to support use of CAC and/or FORTEZZA for storing the user's certificates appropriate for the particular application. These APIs include RSA Cryptoki or Microsoft CryptoAPI for CAC, Cryptoki or Microsoft Cryptographic Service Provider plug-in for FORTEZZA).		Verify that the browser provides the necessary APIs and/or plug-ins to accommodate use of CAC and/or FORTEZZA (as appropriate).	V1	3 (PKI Tokens)	
4.4.27: I&A through encrypted HTML forms	I&A implemented based on UserID and static password, should be implemented with UserID and password transmitted from browser to server either via (1) encrypted HTML form fields (never unencrypted), or (2) encrypted one-time cookies (never unencrypted or persistent).	The application is a Web application intended to serve users for whom the requirement to use PKI certificates has been waived	Verify the requirement to perform I&A via encrypted HTML forms was designed into the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by testing.	V5, V18, V19	BP	
4.4.28: Non-PKI I&A	The application must use PKI-based I&A, and must not use HTML forms to authenticate clients except when the application is designed to be used by users for whom the requirement to use PKI certificates has been waived (i.e., DOD/military retirees and dependents; academia)	The application is a Web server application	Verify requirement to disallow the use on non-PKI I&A was designed into the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by testing. If not, verify that an exemption waiver is on hand.	V1	BP	
4.4.29: No basic authentication	The Web server's Basic authentication capability must not be used over connections that are not protected by HTTPS and SSL/TLS.	The application is a Web application	Verify that the application cannot authenticate over standard HTTP connections (non-encrypted).	V1, V5	BP	

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.4.30: No hard-coded credentials	The application must not include hard coded credentials stored within or mapped to a Web page, script, function key, or any other type of source code file.	The application is a Web application	Verify the application does not use hard-coded credentials. Verification can be performed scanning the application to search for credentials and other sensitive information.	V1, V2, V5	2d (IAIA-2), 2e (IAIA-2)	
4.4.31: No I&A by Java applets	Web applications must not use Java applets to perform I&A.	The application is a Web application.	Verify that the application I&A is not implemented via Java applets.	V1	BP	V1.1: 4.1.15
4.4.32: Required certificate types	The following certificate types must be used for Web server applications: (1) Unclassified private Web server: Class 3 or Class 4 PKI (X.509) certificates transmitted via Secure Sockets Layer (SSL); (2) Public-access Web server: Class 3 or Class 4 PKI (X.509) certificates transmitted via SSL; (3) Classified Web server: appropriate class of PKI (X.509) certificates as determined by the classification of the server, and transmitted via SSL.	The application is a Web Server application	Verify that the certificates used for I&A to the Web server application are of the appropriate Class, and are transmitted via SSL.	V1	3 (Web server access control via PKI)	V1.1: 4.1.11, 4.1.12, 4.1.13
4.4.33: Class 4 certificates	The application must be able to accommodate use of Class 4 certificates with minimal change to the application code.	The application currently performs I&A using Class 3 certificates	Verify that the I&A mechanism can accommodate Class 4 certificates with little or no change to application code.	V1	3 (Evolution of DOD certificates)	V1.1: 4.1.16
4.4.34: I&A using biometrics	The application I&A mechanism shall use biometrics in accordance with DOD policy.	The application performs user I&A using biometrics	Verify that the application I&A mechanisms that use biometrics are implemented in compliance with DOD policy, when such a policy is available.	V1	1 (4.8.2), 13, 29 (3.2.1.4.4)	V1.1: 4.1.18 As of February 2003 the draft DOD biometric policy has not yet been approved and finalized.
4.4.35: Strong passwords	The application's password management mechanism must prevent users from choosing passwords that do not comply with the password construction rules defined in DODD 8500.1, i.e., (1) The password must be case-sensitive; (2) The password must contain at least eight characters; (3) The password must not contain spaces or a "+"; (4) The password must contain at least one [1] uppercase letter, one [1] lowercase letter, and one [1] non-alphanumeric ("special") character. In addition, the password should not constitute or contain: (1) a word found in the dictionary of a major human language (e.g., English, French, German, Spanish, 2) a text string commonly known to be used as a password (e.g., "password", "administrator", "nobody"), (3) a string(s) of repeating characters, e.g., "ee", designated by the administrator as prohibited, (4) the user's name or user ID	The application performs user I&A based on UserID and static password	Verify that password management mechanism rejects user-selected passwords that do not conform to specified password construction rules.	V1, V4	2d (IAIA-2), 2e (IAIA-2); 29 (3.2.1.4.1.6, 3.2.1.4.1.7, 3.2.1.4.1.8)	V1.1: 4.1.19, 5.1.1 An example of a correctly constructed password: C@5t1e!
4.4.36: Assigment of User and Group IDs	The application must not prevent the administrator from assigning any UserID he/she chooses to any user account, or from assigning any GroupID he/she chooses to any group account. The application mu st not force the administrator to assign a particular UserID to a particular user account (e.g., "Administrator" to the administrator account), and must not force the administrator to assign a particular GroupID to a particular group account.		Verify that the application does not constrain administrator's assignment of UserIDs to accounts.	V1	BP	V1.1: 4.1.34

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.4.37: Trustworthy credentials only	The application must not authenticate users based on UserID alone; the application must require users to present a trustworthy authentication credential (e.g., password, certificate, and biometric).	The application performs user I&A based on UserID and static password	Verify that the application does not allow login by users who do not present trustworthy authentication credentials.	V1	BP	V1.1: 4.1.34
4.4.38: Password changes by user	The application's password management mechanism must: (1) Enable the administrator to assign passwords to users; (2) Require the user to change his/her administrator-assigned password after the first login using that password; (3) Enable the user to change his/her own password on demand thereafter with no restrictions as to frequency of changes allowed, (4) Require that the new password selected by the user contain at least four [4] new characters.	The application performs user I&A based on UserID and static password	Verify that the application's password management mechanism enables administrators to assign user passwords, requires users to change administrator-assigned passwords, and enables users to change their own passwords on demand with no restrictions as to frequency of changes allowed.	V1, (V4)	2d (IAIA-2), 2e (IAIA-2); 29 (3.2.1.4.1.1, 3.2.1.4.1.5)	V1.1: 4.1.20, 5.1.2
4.4.39: Password changes by administrator	The application must ensure that only the administrator is allowed to change passwords other than his/her own.	The application performs user I&A based on UserID and static password	Verify that the application does not allow users other than the administrator to change passwords not associated with their own UserIDs.	V1, (V4)	29 (3.2.1.4.1.4)	V1.1: 4.1.20
4.4.40: New password after expiration	The application must not authenticate a user whose password has expired until the user changes the expired password.	The application performs user I&A based on UserID and static password	Verify that the application forces a user whose password has expired to select a new password before authenticating him/her.	V1	29 (3.2.1.4.1.1.2)	V1.1: 4.1.22 This requirement is optional for applications that handle only publicly-releasable data.
4.4.41: Non-reuse of expired password	The application's password management mechanism must be able to recognize a user's attempt to choose one of his/her previous, now-expired password(s), and must prevent the user from choosing such a password. The new password chosen by the user must contain at least four (4) characters not found in the user's expired password. The administrator should be allowed to specify the number of expired previous passwords that must not be chosen by a user.	The application performs user I&A based on UserID and password	Verify that the application prevents the user from choosing one of his/her previous, expired passwords. Verify that the application allows the administrator to specify how many previous passwords cannot be chosen by the user.	V1	2d (IAIA-2), 2e (IAIA-2); 29 (3.4.1.4.1.3)	V1.1: 5.1.3
4.4.42: Unique User IDs	The application must not allow the same UserID to select or enter more than one password.	The application performs user I&A based on UserID and password	Verify that the application does not allow one UserID to choose or login using more than one password.	V1	2d (IAIA-2), 2e (IAIA-2)	V1.1: 4.1.30
4.4.43: Unique passwords	The application must not allow more than one UserID to select or enter the same password.	The application performs user I&A based on UserID and password	Verify that the application does not allow more than one UserID to choose or login using same password.	V1	2d (IAIA-2), 2e (IAIA-2)	V1.1: 4.1.31
4.4.44: Password expiration	The application's password management mechanism must enable the administrator to set an expiration threshold for every password associated with every UserID.	The application performs user I&A based on UserID and static password	Verify that the application's password management mechanism enables the administrator to configure a password expiration threshold for every password.	V1	29 (3.2.1.4.1.1)	V1.1: 4.1.21, 5.1.3 This requirement is optional for applications that handle only publicly-releasable data.
4.4.45: Authentication for every session	The application must require the user to type his password every time he attempts to initiate a new processing session. The application must not store user passwords in cookies, client- or server-side scripts, or any other "replayable" form that automates user login so that the user does not have enter his password to login when initiating a new	The application performs I&A based on UserID and static password	Verify that the application requires the user to log in every time he initiates a session. Verify requirement was designed into the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by testing.	V1, (V19)	BP	

	session.					

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.4.46: No anonymous accounts	The application must not authenticate anonymous UserIDs.	The application performrs user I&A based on UserID and password	Verify that the application does not allow login by anonymous UserIDs.	V1	BP	V1.1: 4.1.32
4.4.47: Explicit log-out	The application must enable the user to explicitly log out to terminate his/her session.	Application performers user I&A based on UserID and password	Verify that the application allows the user to explicitly terminate a session.	V33	BP	
4.4.48: Assurance of I&A mechanisms	The application may use an I&A technology other than UserID and static password only if that I&A technology can be proved to be at least as secure as UserID and static password I&A	The application is an electronic records management application	Verify that it is not as easy to bypass or subvert the alternate I&A mechanism as it is to subvert (e.g., through password guessing or cracking) authentication based on UserID and static password.	V1	6 (C2.2.7.1)	
4.4.49: Confidentiality of transmitted I&A data	The application must encrypt user passwords and any other sensitive I&A data before transmission over a network, and the strength of that encryption must be at least equivalent to the assurance and robustness of encryption used to protect the information that will be accessed after the user is authenticated (i.e., if the information is classified and transmitted over a lower-level network, the password must be encrypted with at least NSA-approved Type 1 high robustness encryption).	Application transmits sensitive I&A data (e.g., passwords, biometric data, certificates) over a network	Verify that the application encrypts user passwords with appropriate, approved cryptographic technology of the appropriate level of assurance and robustness before transmitting them over network.	V1, V5, V18	2d (IAIA-2), 2e (IAIA-2); 27 (6.6.4.5); 29 (3.2.1.5.1)	V1.1: 4.1.24, 5.1.4 Use of hexadecimal or another non-cryptographic encoding scheme instead of encryption is unacceptable.
4.4.50: Confidentiality of password in use	The application must prevent any other process or user from reading cleartext passwords while they are being used by the application.	Application manipulates cleartext passwords.	Verify that during the process of manipulating cleartext passwords, the application prevents any other process or user from reading the cleartext I&A data.	V1, V5	BP	V1.1: 4.1.25
4.4.51: Confidentiality of I&A data at rest	The application must ensure that user passwords and all other sensitive I&A data (certificates, biometric templates, raw biometric material, etc.) used by the application are encrypted before they are stored. The strength of cryptography used must be least equivalent to the assurance and robustness of encryption used to protect the information that will be accessed after the user is authenticated when that information is at rest. Furthermore, the application must not be exploitable by unauthorized users to decrypt and read the application users' stored I&A.		Verify that the application ensures that all I&A data it uses are encrypted with appropriately assured, robust cryptography before those data are stored. Verify that the application cannot be used in any way to decrypt and gain unauthorized read-access to the I&A data used to authenticate users of the application.	V1, V5	2d (IAIA-2), 2e (IAIA-2); 29 (3.2.1.5.2)	
4.4.52: Integrity of I&A data	The application must not be exploitable to modify I&A data such as passwords, certificates, biometric templates, raw biometric material, etc.		Verify that the application cannot be used to bypass access controls or spoof trusted users to gain unauthorized write-access to the I&A data used to authenticate users of the application.	V1, V2, V6	15 (4.7-4.8)	V1.1: 4.1.26, 4.4.19
4.4.53: Availability of I&A data	The application must not be exploitable to delete I&A data such as passwords, certificates, biometric templates, raw biometric material, etc.; or to destroy the interface between the application's I&A mechanism and its I&A data. The application's operation must not threaten the availability of the I&A data used to authenticate its users.		Verify that the application cannot be used to bypass access controls or spoof trusted users to gain unauthorized delete-access to the I&A data used to authenticate users of the application, or to attack the interface between the application's I&A mechanism and its I&A data. Verify that the application does not operate in a way that threatens or causes denial of access to the I&A information.	V2	BP	

4.7 Authorization and Session Control

This subsection lists requirements governing how applications authorize users and external processes access to application resources, and to data handled by the application, and also how applications perform session control (i.e., deauthorization and reauthorization of users). An application may invoke an external authorization mechanism, such as a Role Based Access Control implementation, via trustworthy calls, only if that external authorization mechanism is implemented by an approved technology. See IATF (Reference 27) Section 4.3.1.2 for a discussion of the processes and technologies involved in authorization.

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.5.1: User authorization	The application must ensure that users have been authorized to perform the functions they attempt to perform or access the resources (including data) they attempt to access, and that those authorizations explicitly allow them to perform those functions or access those resources/data in the ways the users attempt to do so.		Verify that the application ensures that users have been authorized before allowing them to access the application functions or resources they request.	V2, V34	2d (IAAC-1, PRINK-1), 2e (IAAC-1, PRNK-1), 2f (IAAC-1, PRNK-1)	V1.1: 4.2.1
4.5.2: Authorization management tool	The application must provide a tool for creating and modifying authorization information (e.g., ACLs, active accounts). For server applications, the tool must be able to create or modify this information without having to restart the application. The application must ensure that the tool can be accessed only by an authorized user.		Verify that the application provides a tool for creating and managing authorization information.	V34, (V2)	2d (IAAC-1), 2e (IAAC-1), 2f (IAAC-1); 29 (3.2.4.3, 3.2.16.1, 3.2.16.2, 3.2.16.11, 3.2.17.3, 3.2.17.4)	V1.1: 4.2.2
4.5.3: Interprocess Authorization	The application must also perform interprocess authorization using technology (e.g., X.509 certificates) approved by NSA or NIST and appropriate for the application's Mission Assurance Category.	The application performs interprocess I&A	Verify that the application processes perform interprocess authorization using approved technology.	V34	2a (DCNR-1), 2b (DCNR-1), 2c (DCNR-1)	V1.1: 4.2.6
4.5.4: Application least privilege	The privileges granted to application executables at any point in time (including programs, processes, scripts, Java applets, etc.) must be the absolute minimum privileges required for the executable to operate correctly at that point in time.		Verify that the application is always granted the least privileges necessary to function. It probably will be necessary to verify that this requirement was considered and met during the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by conducting 3rd party source code review.	V26	2d (ECLP-1), 2e (ECLP-1), 2f (ECLP-1); 29 (3.2.15.3)	V1.1: *5.2.5, 5.2.6, 5.2.7*
4.5.5: PBAC or RBAC	The application must implement Policy-Based Access Control (PBAC) or Role-Based Access Control (RBAC) for authorizing user privileges in conjunction with its discretionary and mandatory data access control schemes.		Verify that the application's authorization function is implemented via PBAC or RBAC, and that it is possible to designate access control privileges by role or other policy-determined grouping of users.	V2, V34	2a (ECPA-1), 2b (ECPA-1), 2c (ECPA-1); 27 (4.3.1.3); 29 (3.2.16.1)	
4.5.6: RBAC for privileged accounts	The application must implement RBAC to designate and authorize privileged accounts (e.g., administrator accounts).		Verify that the application ensures that RBAC is used to designate and authorize privileged accounts.	V2, V34	2a (ECPA-1), 2b (ECPA-1), 2c (ECPA-1); 29 (3.2.16.1, 3.2.16.10)	V1.1: 4.2.7, *5.2.1, 5.2.2*

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.5.7: Application roles and privileges	The role(s) assigned to an application process must directly correspond to the duties/functions assigned to that process.	The application implements RBAC	Verify that the roles granted to application processes are correct. It probably will be necessary to verify that this requirement was considered and met during the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by conducting 3rd party source code review.	V34	2d (ECLP-1), 2e (ECLP-1), 2f (ECLP-1)	V1.1: 4.0.12, *5.2.1, 5.2.2, 5.2.4, 5.2.5, 5.2.6, 5.2.7*
4.5.8: RBAC in classified applications	The application must ensure that its RBAC implementation enforces separation of duties and least privilege.	The application implements RBAC and handles classified data	Verify that the classified application ensures that RBAC enforces separation of duties and least privilege.	V2, V26, V34	2d (ECLP-1), 2e (ECLP-1), 2f (ECLP-1)	V1.1: 4.2.8
4.5.9: Process least privilege	Application processes that act on behalf of users or other processes must not be granted privileges greater than those granted to the users or other processes on whose behalf those processes operate.	The application is not a "trusted" (multilevel secure) application	Verify that the application includes no processes that require privileges greater than the privileges granted to the user or other process on whose behalf the processes operate.	V26	2d (ECLP-1), 2e (ECLP-1), 2f (ECLP-1);	V1.1: 4.0.12
4.5.10: No privilege-authorization mismatch	The application's authorization mechanism must prevent unauthorized users from assigning or changing access privileges assigned to user, group, or role.		Verify that only the administrator or other designated trusted user is able to assign or change the access privileges associated with any user, group, or role.	V2	6 (C2.2.3.15); 29 (3.2.16.9)	
4.5.11: Group privileges	The application must enable the creation of different user groups, and the assignment of different privileges to each group.	The application is an electronic records management application	Verify that the application's authorization mechanism supports the creation of user groups, and the assignment of unique privileges to each created group.	V2	6 (C.2.2.7.3)	
4.5.12: Relinquishing of privileges	A privilege should be granted to an application process only for as long as it takes the process to perform the action for which it requires the privilege. The privilege must be relinquished by the process as soon as the process has completed the privileged action. If the process requires the same privilege later to perform another action, that privilege must be granted again in a separate transaction. The process must not "hold onto" any privilege in anticipation of future use.		Verify that the application relinquishes any additional privileges as soon as necessary. It probably will be necessary to verify that this requirement was considered and met during the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by conducting 3rd party source code review.	V26, V33	2d (ECRC-1), 2e (ECRC-1)	
4.5.13: Maximum number of sessions	The application must enable the administrator to configure the maximum number of simultaneous sessions allowable per UserID, role, and per organization/group. The application must also prevent users who reach their maximum number of allowed sessions from initiating another session until they terminate an active session.	The application allows multiple simultaneous sessions by a single user account, role, or group	Verify that the application enables administrator to configure maximum number of simultaneous sessions allowed per UserID, per role, and per organization/GroupID. Verify that the application prevents users who reach this maximum from initiating a new session until they terminate an active session.	V26, V34, (V27); 2d (SCLO-2), 2e (SCLO-2)	BP	V1.1: 4.2.9
4.5.14: Session inactivity timeout	The application must enforce a session timeout that suspends user access to the application after a configured period of inactivity. This session inactivity timeout must not be omitted from the application even if the application implements other periodic timeouts unrelated to inactivity (e.g., to impose arbitrary session lengths). After the timeout occurs, the application must require the user to reauthenticate himself/herself before allowing that user to resume the suspended session. The application must also enable the user to suspend his/her application session at will.		Verify that the application suspends user access after configured period of inactivity. Verify that the application requires user to login again before resuming suspended session. Verify that the user can suspend his/her session at will.	V2, V33	29 (3.2.5.12)	V1.1: 4.2.10 Also referred to as a "deadman" capability.

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.5.15: Configuration of inactivity timeout	The application must allow the administrator to configure the session inactivity timeout.		Verify that administrator can configure duration of inactive session timeout period	V2, V33	29 (3.2.5.12.3, 3.2.5.12.4)	
4.5.16: Session timeout notification	The application's session timeout capability must notify the administrator when a session timeout occurs.		Verify that the administrator is notified whenever an active application/user session times out.	V35	29 (3.2.5.12.5)	
4.5.17: Confidentiality of authorization data	The application must protect the confidentiality of its information associated with user authorization (e.g., access control lists).		Verify that the authorization mechanism used by the application protects confidentiality of the authorization information (e.g., access control lists, etc.) used to make access control decisions governing access to the application's data, executables, and resources.	V2	BP	V1.1: 4.2.3
4.5.18: Integrity of authorization data	The application must ensure that the integrity of the authorization information (e.g., access control lists) used to authorize its users is protected unauthorized modification or substitution.	The application performs authorization of users or processes	Verify that the application adequately protects its authorization information from unauthorized modification or substitution.	V2	BP	V1.1: 4.2.4
4.5.19: Availability of authorization data	The application must not be exploitable to delete authorization data such as access control lists, or to destroy the interface between the application's authorization mechanism and its authorization data. The application's operation must not threaten the availability of the authorization data used to assign privileges and make access control decisions related to its users.	The application performs authorization of users or processes.	Verify that the application cannot be used to bypass access controls or spoof trusted users to gain unauthorized delete-access to the authorization data used to assign privileges and make access control decisions related to users of the application, or to attack the interface between the application's authorization mechanism and its authorization data. Verify that the application does not operate in a way that threatens or causes denial of access to the authorization information.	V2	BP	V1.1: 4.2.5

4.8 Access Control

This subsection lists requirements governing how applications control access by users and external processes to application resources, and to data handled by the application. These requirements pertain both to applications that interact with the underlying host or surrounding infrastructure to provide access control, and to applications that perform their own access control in some form (e.g., using embedded digital rights management mechanisms). See IATF (Reference 27) Section 4.3.1.4 for a discussion of the processes and technologies involved in access control.

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.6.1: Unauthorized access	The access controls used by the application must prevent unauthorized users from reading or manipulating data, application resources, devices, etc. that are created, manipulated, or used by the application.		Verify that the access controls of the underlying host environment are correctly configured, and that all encrypted files are unable to be decrypted, to prevent an account set up without the appropriate privileges and without access to the necessary cryptokey(s) from reading, writing, executing, deleting, copying, or any of the following items created by, belonging to, or used by the application: (1) data files, (2) executable files, (3) devices, 4) configuration files.	V2	2 (5.10.2); 2a (ECCD-2), 2b (ECCD-2), 2c (ECCD-1), 2d (ECCD-1); 6 (C2.2.5.2, C2.2.5.4); 29 (3.2.5)	

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.6.2: Unauthorized actions	The application must prevent authorized users from using the application to perform any function that they are not authorized to perform.		Verify that it is not possible for an account set up without the appropriate privileges to use perform any application function that should not be accessible to that account, given its lack of appropriate privileges.	V2	2 (5.10.2); 6 (C2.2.5.4, C2.2.6.3.2, C2.2.6.4.)	
4.6.3: Unauthorized access to roles	The application must prevent users from performing any functions that are not explicitly authorized for their role(s).		Verify that it is not possible for an account set up to be excluded from a given role to perform application functions that are authorized to be performed only by accounts that belong to that role.	V2	2 (5.10.2); 6 (C2.2.7.2)	
4.6.4: Mandatory access control, classified data	The application must provide the necessary APIs to an underlying OS (and, if appropriate, DBMS) that implements Mandatory Access Controls (MAC) robust enough to protect the classified data from unauthorized disclosure, or use NSA-approved Type 1 cryptography to encrypt the data before storage.	The application stores classified data, and can be accessed by users who are not cleared to read those data	Verify that the application uses underlying OS/DBMS/Web server MACs to protect the classified data, and that it is imp ossible for an account set up with a lower clearance level than required to access data at a particular classification level to access those data. Verify that if the application does not use the underlying MACs to provide this protection, it encrypts the classified data before storage using NSA-approved Type 1 cryptography.	V2		V1.1: 4.2.11, 5.2.8 Applies to all classified applications, regardless of Mission Assurance Category.
4.6.5: Discretionary access control, classified data	The application must provide the necessary APIs to an underlying OS (and, if appropriate, DBMS or Web server) that implements Discretionary Access Controls (DAC) robust enough to protect the data from unauthorized disclosure.	The application stores classified data, and can be accessed by users who do not have a need to know for that data	(1) Verify that the application uses underlying OS/DBMS/Web server DACs to protect the classified data, and that it prevents an account set up not to belong to the role or group (or other privileges indicating need to know) that is authorized to access a particular data item from accessing that data item. (2) Verify that if the application does not use the underlying DACs to provide this protection, it encrypts the classified data before storage using NSA-approved Type 1 cryptography.	V2		Applies to all classified applications, regardless of Mission Assurance Category.
4.6.6: DAC access levels	The application must provide the necessary APIs to an underlying OS and Web server that implements DAC that can support, at a minimum, three levels of access: (1) Open access (no I&A required, 2) Controlled access (requires individual I&A, 3) Restricted access to specific community of interest (requires need to know)	The application is a Web server application	Verify that the access controls used by the application support at least three different access levels, and that the appropriate I&A is required before a user is granted access to a particular level. Verify that the access controls used by the application to separate and protect Restricted Access data prevent any account set up to be outside of the community of interest (as defined by role or user group) from accessing those data.	V2	2d (ECAN-1), 2e (ECAN-1)	
4.6.7: Access control, sensitive and MAC I unclassified data	The application must provide the necessary APIs to an underlying OS (and, if appropriate, DBMS) that implements DACs robust enough to protect the sensitive and MAC I unclassified data from unauthorized disclosure, or use 3 Data Encryption Standard (3DES) or Advanced Encryption Standard (AES) to encrypt the data before storage.	The application stores sensitive and/or MAC I unclassified data and can be accessed by users who are not authorized to read those data	Verify that the application uses underlying OS/DBMS DACs to protect the sensitive and MAC I unclassified data data, or invokes encryption of those data before storage using AES or 3DES cryptography.	V2	BP	V1.1: 4.2.12

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.6.8: Access control, need-to-know separation	The application must provide the necessary APIs to an underlying OS (and, if appropriate, DBMS) that implements DACs robust enough to enforce the separation of data that exist with different needs-to-know, regardless of the classification or sensitivity of those data. If the underlying OS/DBMS does not provide the appropriately robust DACs, the application must provide APIs to a cryptographic module that can be used to encrypt the data before storage to enforce need-to-know separation.	The application stores data with different needs-to-know, and can be accessed by users who are not authorized to view all of those data.	Verify that the application uses underlying OS/DBMS DACs to protect data, or encrypts classified data before storage using NSA-approved Type 1 cryptography.	V2	2d (ECAN-1), 2e (ECAN-1)	
4.6.9: Labeling of Classified data	The application must apply (or allow the user to apply, as appropriate) the appropriate confidentiality and integrity labels to the data at the time of creation or modification. These labels must be understood by the access control mechanism used to control access to the data. NOTE: "Sensitive" in this case is defined in Section 20 of the NIST Act, Title 15 of the U.S. Code Section 278g-3.	The application is used to create or modify classified data	Verify that the application uses underlying OS/DBMS access controls to protect classified data, or encrypts classified data before storage using 3DES or AES.	(V2)	2d (ECML-1), 2e (ECML-1); 29 (3.2.6.1.1, 3.2.6.4, 3.2.7.1)	V1.1: 4.2.14, 5.3.1 Applies to all classified applications, regardless of Mission Assurance Category.
4.6.10: Labeling of Not Publicly Releasable data	The application must apply a label to the data upon creation or modification that clearly indicates that the data are not releasable to the public. These labels must be understood by the access control mechanism used to control access to the data.	The application is used to create or modify private unclassified data	Verify that the application correctly labels classified and sensitive data when those data are created or modified. Verify that the labels are understood by the access control mechanism.	(V2)	2d (ECML-1), 2e (ECML-1); 29 (3.2.7.1)	V1.1: 4.2.15, 5.3.2
4.6.11: Classification labels in metadata	The application must provide a capability to allow users to define appropriate metadata/tags indicating the classification label of a classified data element at the time of creation or modification of that data element. Data elements in this context include whole databases, individual rows, and individual records.	The application is a database application	Verify that the application allows a user to assign metadata or a metatag indicating classification to a complete database, to an individual row in a database, and to an individual record in the database	(V2)	2d (ECML-1), 2e (ECML-1); 6 (C4.1.1); 29 (3.2.18.5)	
4.6.12: Marking of output	The application must ensure that the data are marked to reflect the sensitivity level/classification of data produced by the application (including handling caveats, code words, and dissemination control markings). The application shall provide a capability to enable trusted users to configure the markings to be applied to the application's printed and transmitted output.	The application transmits data or sends it to a printer	Verify that data transmitted or printed by the application are appropriately marked.	(V2)	2d (ECML-1), 2e (ECML-1); 29 (3.2.8.3, 3.2.8.4, 3.2.8.5, 3.2.8.6, 3.2.8.7, 3.2.8.8, 3.2.16.6)	V1.1: 4.2.16
4.6.13: Invalid pathname references	Whenever a pathname or URL referenced in the application code is changed or removed from the system, the application code must be changed to change or delete that reference.	The application is a Web application	Verify that the application code does not include any references or pointers to nonexistent pathnames or URLs.	V14	BP	V1.1: 4.2.17
4.6.14: Truncated pathnames	If a user presents a truncated pathname or URLs that do not end in a file name: The application must not allow the user to access the file system directory indicated by the pathname.	The application is a Web application	Verify that the application does not accept truncated pathnames from users.	V13	BP	V1.1: 4.2.18
4.6.15: Relative pathnames	The application's references to pathnames and URLs must point to the absolute pathname/URL, not to a relative pathname or URL.	The application is a Web application whose code contains references to pathnames or URLs	Verify that the application code does not contain references to relative pathnames or URLs.	V12	BP	V1.1: 4.2.19
4.6.16: User input of relative pathnames	The application must not accept relative pathnames or URLs input by users.	The application is a Web application	Verify that the application does not accept relative pathnames or URLs input by users.	V12, (V8)	BP	V1.1: 4.2.20

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.6.17: No directly-entered URLs	The application must not allow users to access Web pages/resources not explicitly allowed by links on the portal page by directly typing the URLs of the forbidden pages/resources into their browser's "Location" line.	The application is a Web portal application	Verify that the application does not allow users to access pages/resources not explicitly allowed by links on the portal page.	V20	BP	V1.1: 4.2.21
4.6.18: Protection of user identity	Browsers and other client applications should ensure that cookies and other user identity information stored on the browser/client platform are protected from disclosure and tampering.	The application is a Web client	Verify that the browser prevents unauthorized reading or tampering with stored cookies and other user identity information.	V2, V19	BP	V1.1: 4.2.22
4.6.19: CGI script "holes"	CGI scripts must not contain "holes" that can be exploited to gain direct access to the underlying operating system or to otherwise compromise the application.	The application is a Web application	Verify that the any CGI scripts contain no exploitable "holes."	V16	BP	V1.1: 4.2.23 Review the description of Vulnerability 16 and the for more information on CGI script "holes". Review the Application Security Developer's Guide for information on how to identify and avoid CGI script "holes" of many types.
4.6.20: Database views	The application must not rely on database views as an access control mechanism.	The application is a database application	Verify the application does not rely on database views as an access control mechanism. Verification can be accomplished by reviewing the database's access control settings	V2	BP	
4.6.21: Data change notification	The application must indicate to users, upon access to data or file, the date and time of the most recent change to those data.		Verify that the application displays a notification message that informs the user who accesses a file or data item of the date and time of the most recent change to that file/data item.	(V2)	2a (ECCD-2), 2b (ECCD-2), 2c (ECCD-1), 2d (ECCD-1)	V1.1: 4.2.13
4.6.22: Unauthorized metadata changes	The application must prevent unauthorized users from changing any metadata associated with database entries or records created/manipulated by the application.	The application is a database application	Verify that the access controls used by the application prevent an account created without the necessary privileges from changing the metadata associated with a database entry and with a database record.	V2	2a (ECCD-2), 2b (ECCD-2), 2c (ECCD-1), 2d (ECCD-1); 6 (C2.2.3.13, C2.2.3.16, C2.2.3.22)	
4.6.23: Changes to record associations	The application must prevent unauthorized users from changing or deleting any established reference links, or associations, or other relationships between data elements.	The application is a database application	Verify that the access controls used by the application prevent an account created without the necessary privileges from changing the established links, associations, and other relationships between data elements in the database.	V2	6 (C2.2.3.17)	
4.6.24: No modification of read-only data	The application must prevent modification of any data that are designated as read-only. The application should also issue a warning reminding the user that the data are read-only when that user attempts to move or delete read-only data.		Verify that the access controls used by the application prevent any account, no matter what privileges are assigned to that account, from modifying or overwriting read-only data. Verify that when a user attempts to move or delete read-only data, the application notifies the user that the data are read-only.	V2	6 (C2.2.4.2)	

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.6.25: No unauthorized access to cryptographic materials	The application must ensure that its access controls prevent unauthenticated, unauthorized users from gaining access to any cryptographic material used by the application, including keys, trust points, and certificates. The application must also ensure that access to private keys is strictly limited to users authorized to access those keys.	The application uses cryptography	Verify that the access controls used by the application prevent an account set up with insufficient privileges from accessing any of the cryptographic material (e.g., keys, trust points, certificates) used by the application. Verify that access controls used by the application prevent an account set up with any identity but that of the user who owns a private key—regardless of the privileges assigned to that account—from accessing the other user's private key.	(V2)	2a (ECCD-2), 2b (ECCD-2), 2c (ECCD-1), 2d (ECCD-1); 16 (4.2.3)	V1.1: *4.3.8*
4.6.26: Separation of encrypted and unencrypted data	The application must ensure that its access controls maintain strict separation between encrypted and unencrypted data created or manipulated by the application.	The application uses encryption	Verify that the application stores encrypted data with different access control characteristics (e.g., MAC label, DAC privileges, etc.) than the data it stores in the clear.	(V2)	2d (ECNK-1), 2e (ECNK-1); 16 (4.2.3)	
4.6.27: Cryptographic separation of classified transmissions	The application must invoke NSA-approved Type 1 encryption of each data stream to accomplish mandatory separation of different classifications of data.	The application transmits more than one level of classified data, or a mixture of classified and SBU or Unclassified data over the same network.	Verify that when the application is going to transmit data at one classification level, it invokes NSA-approved Type 1 encryption of that data using a key designated for that classification level before transmitting the data, and that when it is going to transmit data at a different classification level, it invokes NSA-approved Type 1 encryption of the data with a different key designated for that second, different classification level.	V2	16 (4.2.3)	V1.1: *4.3.3*
4.6.28: Cryptographic need to know separation of transmissions	The application must invoke NIST FIPS 140-1 certified encryption of each data stream to accomplish separation of different need to know compartments or categories of data transmitted over the same network.	The application transmits more than one need-to-know compartment or category of data over the same network. No waiver has been granted to allow different compartments/ categories of data to be transmitted over the same network without cryptographic separation.	Verify that when the application is going to transmit data at one need-to-know, it invokes certified NIST FIPS-140 encryption of that data using a key designated for that need to know before transmitting the data, and that when it is going to transmit data with a different need to know, it invokes FIPS 140-1 encryption of the data with a different key designated for that second, different need to know.	V2	2d (ECNK-1), 2e (ECNK-1); 16 (4.2.3)	
4.6.29: Cryptographic separation of SAMI transmissions	The application must invoke NSA-approved Type 1 encryption of any SAMI data stream transmitted over the same network with non-SAMI data.	The application transmits more than SAMI data over a network at the same classification level as the data.	Verify that when the application is going to transmit SAMI data, it invokes NSA-approved Type 1 encryption of the data before transmission.	V2	2d (ECNK-2); 16 (4.2.3)	
4.6.30: Configurable access controls	The access control mechanism used by the application must provide an interface or tool to enable the administrator to define the access control characteristics of each data object and resource to be controlled with relation to the individual user, group, role, etc. that is allowed to access it.		Verify that the access control mechanism used by the application enables the administrator to define the access control characteristics of all data objects, executable files, configuration files, etc. owned or used by the application. These characteristics should be definable in terms of the access privileges to those objects granted to an individual user account, a group account, and a role account.	V2	29 (3.2.5)	

4.9 Confidentiality

This subsection lists requirements governing the methods used by applications to ensure the confidentiality of the data they manipulate, store, or transmit. These requirements pertain to applications which augment, at the application layer—either through embedded functionality or by secure calls to approved external encryption mechanisms—those confidentiality controls provided at lower (e.g., network, data link) layers such as Virtual Private Networks and link encryption. See IATF (Reference 27) Section 4.3.2 for a discussion of the processes and technologies involved in confidentiality.

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.7.1: Encryption API	There must be an API that enables the application to invoke an encryption capability to selectively encrypt data and files.		Verify that an API is present that enables the application to selectively invoke encryption.	V3, V23	BP	V1.1: 4.3.1
4.7.2: Nondisclosure of cleartext data	The application must ensure that sensitive cleartext data are not disclosed before they are encrypted.		Verify that the application ensures that cleartext data are protected from disclosure before they are encrypted.	V1, V5, V18	BP	V1.1: 4.3.2
4.7.3: Encryption before transmission, classified or SAMI data, different level network	The application must invoke NSA-approved Type 1 (high robustness) encryption of the data before transmitting the data.	The application transmits data over a networ, and one or more of the following is true: (1) The data are classified higher than network; (2) Some users on network are not cleared to read data at this classification; (3) The data are SAMI data; (4) The data are classified and the network is a public network.	Verify that the application invokes appropriate, approved encryption technology to encrypt data before transmission (if necessary).	V5	2d (DCSR-3, ECCT-2); 27 (7.1.4.4)	V1.1: 4.3.3 Encryption solves the problem that arises that, when in transit, the data fall outside the protection of the application that generated them.
4.7.4: Encryption before transmission, classified data, different need-to-know network	The application must invoke encryption of data before transmission using 3DES or AES (medium-robustness), or the application owner must get a signed waiver from the data owner allowing the application to transmit the data unencrypted.	The application transmits data over a network. The network and data are at the same classification, but the data have a different need-to-know than the network.	Verify that the application invokes appropriate, approved encryption technology to encrypt data before transmission (if necessary).	V5	27 (7.1.4.4)	V1.1: 4.3.3
4.7.5 : Encryption before transmission, sensitive or MAC I unclassified data, public network	The application must invoke encryption of data before transmission using 3DES or AES (medium robustness). If the data are National Security Data, the cryptography used should use NSA-approved key management.	The application transmits sensitive-but-unclassified or MAC I unclassified data over a public network.	Verify that the application invokes approved medium-robustness encryption technology (AES or 3DES) with appropriate key management to encrypt data before transmission (if necessary).	V5	2e (DCSR-2, ECCT-2); 27 (7.1.4.4); 29 (3.2.21.2, 3.2.21.8)	V1.1: 4.3.3

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.7.6: Encryption of classified or SAMI data at rest	The application must invoke encryption of data before storing those data unless the underlying host provides high-robustness access control and confidentiality protection of the data at the file system or DBMS level (in which case the application need only ensure encryption at a basic level of robustness). If the data are SAMI data, the application must invoke NSA-approved Type 1 encryption. For all other data, the application must invoke NIST FIPS 140-1 certified encryption. If the data are National Security data, NSA-approved key management should be used by the application in connection with encryption of those data.	A waiver has not been granted by the responsible CIO allowing the data to be stored in unencrypted form, and one or more of the following is true: (1) The data are classified, and the application can be accessed by users not cleared to read data of that classification; (2) The data are SAMI, and the application can be accessed by users not authorized to read SAMI data.	Verify that the application invokes appropriate, approved encryption technology to encrypt data before storing them, if necessary.	V2	2d (DCSR-3, ECCR-3); 27 (7.1.4.4)	V1.1: 4.3.4 Encryption solves the problem that arises when the application is not running, and is thus no longer able to control access to the stored data it has generated.
4.7.7: Encryption of MAC I sensitive data at rest	The application must invoke NIST FIPS 140-1 certified 3DES or AES (medium robustness) encryption of data before storing those data. If the data are National Security data, NSA-approved key management should be used by the application in connection with encryption of those data.	A waiver has not been granted by the responsible CIO allowing the data to be stored in unencrypted form. The data are MAC I sensitive data, and the application can be accessed by users not authorized to read MAC I and/or sensitive data.	Verify that the application invokes appropriate, approved medium-robustness encryption technology to encrypt data before storing them (if necessary).	V2	2d (ECCR-2), 2e (DCSR-2, ECCR-1)	V1.1: 4.3.4
4.7.8: Encryption of non-MAC I sensitive data at rest	The application must invoke encryption of data using medium-robustness cryptography before storing those data, unless the underlying host provides medium-robustness access control and confidentiality protection of the data at the file system or DBMS level (in which case the application need only ensure encryption at a basic level of robustness). If the data are National Security data, NSA-approved key management should be used by the application in connection with encryption of those data.	.	Verify that the application invokes appropriate, approved encryption technology to encrypt data before storing them (if necessary).	V2	2d (ECCR-2), 2e (DCSR-2, ECCR-1)	V1.1: 4.3.4
4.7.9: Protection of cryptokeys	The encryption facility invoked by the application must ensure that unauthorized users cannot access the cryptokeys needed to decrypt the data.	The application ensures that data are encrypted	Verify that the encryption facility invoked by the application allows only authorized users to access the cryptokeys needed to decrypt data encrypted by that facility.	V2, V3, V23	BP	V1.1: 4.3.5, 4.4.19
4.7.10: PKI encryption certificates	The PKI invoked by the application must use DOD PKI Class 4 or Class 3 encryption certificates when performing the encryption.	The application invokes a PKI to encrypt data	Verify that the PKI invoked by the application uses DOD PKI Class 3 or Class 4 certificates to perform the encryption.	(V3, V23)	4 (Digitally Signed Email)	V1.1: 4.3.6
4.7.11: Application object reuse	Before shutdown, the application must delete/erase all temporary files, cache, data, and other objects it created during its execution.		Verify that before shutdown, application deletes/erases all temporary objects it created during its execution.	V2	2d (ECRC-1), 2e (ECRC-1); 27 (7.1.5.2.2)	V1.1: 4.3.7

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.7.12: Confidentiality of cryptographic material	The encryption facility invoked by the application must protect from disclosure all sensitive cryptographic material—that is, keying material, private keys, and (if so indicated by the application's robustness) the cryptographic algorithm implementation.		Verify that the encryption facility invoked by the application protects cryptographic data from disclosure.	V2	BP	V1.1: 4.3.8
4.7.13: Integrity of cryptographic functions and material	The application must ensure that all sensitive cryptographic functions and material used by the application are protected from tampering (corruption or modification) by users or processes.		Verify that the application protects all cryptographic material from corruption or unauthorized modification.	V2	16 (4.2.3)	
4.7.14: Cryptokey revocation	The encryption facility invoked by the application must handle and respond correctly to Certificate Revocation Lists (CRLs) and Key Revocation Lists (KRLs) issued by the cryptographic implementation and must not continue to use or accept revoked certificates or keys.		Verify that the invoked encryption facility responds correctly to KRLs and does not use revoked keys.	V23	2d (ECRC-1), 2e (ECRC-1); 16 (4.3.2.4)	V1.1: 4.4.20
4.7.15: Confidentiality of user identities	The application must not: () Reveal to external users or processes the identity of any user associated with any application session; (1) Include within or append onto a data object an indicator of the identity of the data's creator or sender; (2) Invoke any external process that includes within or appends onto a data object any indicator of the identity of the data's creator or sender.	There is an operational requirement for the identities of users to be protected from disclosure.	(1) Verify that the application and any processes it invokes do not automatically include in/append to data any indicator of identity of data's creator or sender. (2) Verify that the application does not reveal the identity of any user associated with any application session.	V2	BP	V1.1: 4.3.9
4.7.16: Protection of sensitive Web transactions	The application should use SSL or TLS (SSL Version 3.0 or TLS Version 1.0) with approved cryptographic and key management algorithms to implement seamless end-to-end session encryption of all network-based Web transactions in which sensitive information is transmitted.	The application is a Web application.	Verify that the application is using an approved cryptographic suite (i.e., FIPS-compliant). Additionally verify that all of the application's cryptologic functions are handled by the approved suite	V5	BP	
4.7.17: No storage of sensitive data in scripts	The application must not store sensitive information of any type in Web scripts.	The application is a Web application	Verify the application is not storing sensitive information in web scripts. Compile a list of sensitive information and scan code (visually and automated) to located the presence of such information.	V2	BP	
4.7.18: HTTP POST for sensitive data	The application should use HTTP POST only, and never HTTP GET, over SSL-encrypted connections to transmit sensitive information, including data in HTML forms.	The application is a Web application	Verify that the application will only transmit sensitive information using HTTP POST commands sent via SSL connections.	V5	BP	
4.7.19: Browser application facilities	Scripts, cookies, or plug-ins should be used in Web client (browser) applications only when the desired functionality cannot be implemented using a more secure mechanism.	The application is a Web client application	Verify that Web client applications use no scripts, cookies, or plug-ins unless it can be proven that no more-secure alternative can be used to achieve the same functional objective.	V31, V19	BP	V1.1: 4.0.14 These mechanisms should not be used just because they are more convenient or familiar to the developer.
4.7.20: Encrypted cookies only for sensitive data	Only encrypted non-persistent (one-time) cookies may be used for transmitting sensitive data. Unencrypted cookies and persistent cookies (encrypted or not) must never be used to transmit sensitive data.	The application is a Web application	Verify that the application does not use persistent or unencrypted cookies to store or transmit sensitive data.	V19	BP	

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.7.21: No storage of sensitive data in immutable Java types	The application must not store passwords and/or other sensitive data in an immutable type (e.g., String) or in any type that has to wait for garbage collection to purge the data from memory. The application should store sensitive data in char[].	The application is a Java application	Verify that the application does not rely on operating system or programming language automated garbage collection to delete data types that store sensitive data. Verify requirement was designed into the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by conducting 3rd party source code review.	V2	BP	
4.7.22: No persistence of files in memory	The application must not contain processes that create temporary files or file copies unless these files/file copies are immediately purged from memory upon termination of the process that created them.		Verify that if the application requires temporary files, that these files are immediately purged upon process termination. Verify that this requirement was designed into the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by conducting 3rd party source code review.	V2	2d (ECRC-1), 2e (ECRC-1)	
4.7.23: No sensitive data in redirects	The application must not include sensitive information in any redirect messages that it returns to clients.	The application is a Web client application	Verify that the application does not include sensitive information in any redirect messages.	V2	BP	
4.7.24: No questionable URL extensions	The application must be able to recognize questionable URL extensions, and validate all URLs sent to it by clients to: () ensure they do not contain such extensions, OR, () truncate the URL to remove the dubious extension.	The application is a Web server application	Verify that the application is able to recognize and validate questionable URL extensions.	V8	BP	
4.7.25: Limit data returned to client	In response to a request or query from a client, the application must return only the data requested, and no additional data.	The application is a Web server application	Verify that the application returns only the data requested, and no additional data. This requirement will probably need to be considered and met during the application's development and subsequently tested by review of code.	V2	BP	
4.7.26: No sensitive data stored by client	The application must not store any sensitive data. All sensitive data should be stored by the Web server, and retrieved by the client only when needed.	The application is a server application	Verify that the application does not store any sensitive data. This requirement will probably need to be considered and met during the application's development and subsequently tested by review of code.	V2	BP	

4.10 Integrity

This subsection lists requirements governing how applications ensure the integrity of the data they manipulate, store, or transmit, as well as the integrity of their own data, executables, and runtime resources. These requirements pertain to applications in environments in which the application augments at the application layer—through embedded functionality or secure calls to approved external integrity mechanisms (e.g., PKI-based cryptographic hash or digital signature mechanisms)—any integrity controls provided by the application's underlying host operating system and surrounding security infrastructure. See IATF (Reference 27) Section 4.3.3 for a discussion of the processes and technologies involved in integrity.

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.8.1: Integrity of transmitted data	The application must use a NIST certified FIPS 140-1 or NSA-approved technology (as appropriate for the application's Mission Assurance Category and robustness level) to implement a hash (e.g., Secure Hash Algorithm One [SHA-1]), checksum, or digital signature (e.g., DSS) of the data before transmission.	The application transmits data over a high level of concern network.	Verify that the application invokes an approved technology appropriate for the Mission Assurance Category to apply a hash, checksum, or digital signature to data before transmitting them.	V35	16 (4.3.3); 27 (4.3.3.1, 7.1.4.3)	V1.1: 4.4.1
4.8.2: Integrity of transmitted code	The application must invoke an approved digital signature technology to digitally sign the code prior to transmission.	The application is used to transmit application code (may include mobile code) over a high level of concern network with inadequately robust security protections.	Verify that the application digitally signs application code using approved digital signature technology before transmitting it over network.	V35	2a (DCMC-1, ECTM-2), 2b (DCMC-1, ECTM-2), 2c (DCMC, ECTM-1)-1; 5 (1.1.3, 1.2.4)	V1.1: 4.4.2 For additional integrity, the application may also invoke an approved technology to apply a hash or checksum to the code before transmission.
4.8.3: Integrity of stored data	The application must invoke NIST-certified or NSA-approved technology (as appropriate for the application's Mission Assurance Category and robustness level) to apply a hash, checksum, or digital signature to the data before storage.	The underlying host environment does not provide access controls sufficiently assured to protect the integrity of stored files/data.	Verify that the application invokes approved technology appropriate for the Mission Assurance Category to apply a hash, checksum, of digital signature to data before storing them.	V2	BP	V1.1: 4.4.3
4.8.4: Integrity mechanism validation	The application must be able to validate the integrity mechanism, and must reject data for which the integrity mechanism validation fails.	The application is used to retrieve stored data or to receive transmitted data that have an integrity mechanism applied to them:	Verify that the application validates integrity mechanisms applied to data, and does not accept data for which the validation fails.	(V23)	16 (4.3.3.1); 29 (3.2.12.1, 3.2.12.2, 3.2.19.1, 3.2.21.12)	V1.1: 4.4.4
4.8.5: Parameter validation	The application must validate parameters before acting on them, and must reject all parameters for which one or more of the following is true: (1) Not formatted as expected; (2) Do not fall within the expected bounds (length, numeric value, etc.)		Verify that the application validates all parameters, and ensures that they do not violate any of the expected rules for parameters.	V8	BP	V1.1: 4.4.5
4.8.6: Notification of acceptable input	The application must inform the user of the expected characteristics of the input—e.g., length, type (alphanumeric, numeric only, alpha-only, etc.), and numeric or alphabetic range.	The application accepts user input.	Verify that the application informs user of the acceptable characteristics of data to be input by the user.	(V8, V10, V11)	BP	V1.1: 4.4.6, 5.5.2
4.8.7: Input validation	The application must validate all data input by users or external processes, and must reject all input for which one or more of the following is true: (1) not formatted as expected; (2) contains incorrect syntax; (3) not a valid data string; (4) contains parameters or characters with invalid values; (5) falls outside the expected bounds (e.g., length, range); (6) contains a numeric value that would cause a routine or calculation in the application to divide any number by zero; (7) contains any parameters the source of which cannot be validated by the user's session token; (8) can induce a buffer overflow; (9) contains HTML; (10) contains special characters, meta code, or metacharacters that have not been encoded (if encoding is allowed); (11) contains direct SQL queries; (12) contains any other type of unexpected content or invalid parameters; (13) contains a truncated pathname reference.		Verify that the application validates all data input by users, and ensures that the input data do not violate any of the expected characteristics for user input.	V7, V10, V11	BP	V1.1: *4.4.5*, *4.4.7*, 5.5.3

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.8.8: Completion of input validation	The application must suspend all processing of the transaction in which input has been received until the input has been completely validated.		Verify that the application was designed to validate all input before processing it. Verify requirement was designed into the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by conducting 3rd party source code review.	V8	BP	V1.1: *4.4.5, 4.4.7*
4.8.9: Argument validation	All application programs, including CGI and shell scripts, must perform input validation on arguments received before acting on those arguments.		Verify that the application was designed to validate all input before processing it. Verify requirement was designed into the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by conducting 3rd party source code review.	V8	BP	V1.1: *4.4.5, 4.4.7*
4.8.10: Validation of external and third-party input	The application must validate all inputs it receives from any external processes, including processes in third-party software integrated into the application, in the same way it validates user input data.		Verify that the application was designed to validate all input from external processes. Verify requirement was designed into the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by conducting 3rd party source code review.	V8	BP	V1.1: *4.4.5, 4.4.7*
4.8.11: Bounds-checking functions only	All functions in the application program must perform bounds checking, such that the functions check the size of all buffer or array boundaries before writing to them, or before allowing them to be written to. In addition, the application must limit the size of what it writes to the buffer or array to the size imposed by the buffer/array boundaries (i.e., to prevent what is written from exceeding the buffer/array size and overflowing the boundary).		Verify that the application was designed to check array boundaries. Verify that that after bounds checking the buffer or array, the application does not write data to that buffer/array that exceeds the size of the buffer/array. Verify requirement was designed into the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by conducting 3rd party source code review.	V7	BP	
4.8.12: Bounds checking on all array and buffer accesses	The application must bounds check all arrays and buffers every time those arrays/buffers are accessed.		Verify that the application was designed to bounds check all arrays and buffers upon access. Verify requirement was designed into the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by conducting 3rd party source code review.	V7	BP	
4.8.13: Input validation before database copying	The application must validate all input data before copying those data into the database.	The application is database front-end	Verify that the application was designed to validate all input before processing it. Verify requirement was designed into the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by conducting 3rd party source code review.	V11	BP	V1.1: *4.4.5, 4.4.7*
4.8.14: No HTML in untrusted input	The application must reject any input containing HTML (including HTTP strings that contain HTML tags) from an untrusted user or other untrusted source.	The application is a Web server application	Verify that the application was designed to validate the source of all input before processing it. Verify requirement was designed into the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by conducting 3rd party source code review.	V1, V34	BP	

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.8.15: Rejection of incorrect input, high and medium robustness	The application process that received the invalid input must gracefully terminate the user process with an error message to the user indicating that the process is terminating as a result of an input error.	The application is of high- or medium-robustness. Data input by users or external processes cannot be validated.	Verify that the application issues an error message to the user warning that the user process is being terminated as a result of an input error, then gracefully terminates the user process.	V15	BP	V1.1: 4.4.8
4.8.16: Rejection of incorrect input, low robustness	The application process that received the invalid input must request the external user or process to reinsert the data. If the reinserted data is still invalid, the application must gracefully terminate the user process with an error message to the user indicating that the process is terminating as a result of an input error.	The application is of low-robustness. Data input by users or external processes cannot be validated.	Verify that the application requests the user or external process to reinsert the data (ideally, with a reminder of acceptable input characteristics). If the application cannot validate the new data, verify that it issues an error message to the user warning that the user process is being terminated as a result of an input error, then gracefully terminates the user process.	V15	BP	V1.1: 4.4.8 A low-robustness application may simply issue the warning and terminate, without allowing the user to attempt to resubmit the data.
4.8.17: Input validations by server only	All user input validations must be performed by the server application even if input validation has already been done by the client application. The client application must not be relied on to perform trustworthy input validation. At best, client input validation can be used to prescreen data before it is validated by the server.		Verify that all validations of user inputs are performed by the server application, even if some input validation has already been done by the client application.	V8	BP	V1.1: 4.4.9
4.8.18: No execution of active content in data	The application's validation of user input data that contains active content (e.g., mobile code) must not result in the execution of the active content.		Verify that application validation of user input that contains active content does not cause that active content to execute.	V9	BP	V1.1: 4.4.10
4.8.19: Process integrity during updates	The application's data update processes must operate correctly, and must not incorrectly reparse, inadvertently introduce errors to, or otherwise corrupt the data they update.	The application updates data.	Verify that the application processes that update data ensure that data updates do not contain errors and do not otherwise corrupt data being updated.	V6	BP	V1.1: 4.4.11
4.8.20: Validation of integrity mechanism on transmitted code	The application must find and validate the digital signature and any hash, checksum, or other additional integrity mechanism applied to that code prior to executing it. If the code's integrity mechanism cannot be validated, or is not present, the application must discard the code without executing it; and audit this discard.	The application receives transmitted executable code (e.g., mobile code):	Verify that the application validates digital signature and any other integrity mechanism applied to application code received over network. Verify that the application discards without execution any code that fails any integrity validation check. Verify that the application discard of code is audited.	V32	29 (3.2.12.1, 3.2.12.2)	V1.1: 4.4.12
4.8.21: Protect server executables from malicious code	The application must invoke a virus scanning tool to scan all files received from users and external processes to ensure these files do not contain malicious content.	The application is a server application whose underlying infrastructure does not adequately protect the application from malicious code.	Verify that the application invokes a virus scanning tool when it first receives a file from a user or external process.	V32	2a (ECVP-1), 2b (ECVP-1), 2c (ECVP-1)	Administrators must keep virus signature files used by virus scanning tools up to date. It is assumed that client applications will be configured with virus scanning as per the relevant STIG.

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.8.22: Protect server configuration from malicious code	The application must invoke a virus scanning tool whenever it executes a program that may access one of the application's configuration or other parameter-containing files.	The application is a server application whose underlying infrastructure does not adequately protect the integrity of the application's configuration and other parameter-containing files from corruption or unauthorized modification by malicious code.	Verify that the application invokes a virus scanning tool when executes a program that may access one of the application's configuration or other parameter-containing files.	V32	BP	V1.1: 4.4.13 Administrators must keep virus signature files used by virus scanning tools up to date. It is assumed that client applications will be configured with virus scanning as per the relevant STIG.
4.8.23: No forwarding of malicious code	The application must ensure that the data to be forwarded do not contain or point to malicious code.	The application is a Web application that forwards data from an untrusted user to another user.	Verify that the application was designed to prevent forwarding of malicious code. Verify requirement was designed into the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by conducting 3rd party source code review.	V32	BP	
4.8.24: Integrity of application executable	The process that validates the application's executable code integrity mechanism checksum or hash must be invoked every time the application is executed to ensure that the application's executable code state has not changed since the original integrity mechanism was applied. If this validation fails, the validation process must prevent the application from being executed, and notify the administrator that the application code needs to be replaced by an uncorrupted executable.	An integrity mechanism (hash, checksum, or other integrity mechanism) was applied to the application's executable code at installation time (prior to the application's first execution).	Verify that the application executable code's integrity mechanism is checked before every application execution. Verify that the integrity checking process prevents execution of applications that fail this integrity check. Verify that integrity checking process notifies administrator when integrity check fails.	V2	BP	V1.1: 4.4.14
4.8.25: Code signing	The application must not execute received code until it: (1) verifies that the code has been digitally signed; and (2) validates the digital signature on the code.	The application receives mobile code, interpreted (versus compiled) code, or other active content.	Verify that the application does not run any mobile, interpreted, or active content code that has not been digitally signed, or for which the digital signature cannot be validated.	V32	BP	V1.1: 4.4.22
4.8.26: Time/date stamp of data modification	The application must time/date stamp each data modification or file update.		Verify that the application applies time/date stamp to data and files each time those data and files are modified.	(V2)	BP	V1.1: 4.4.15, 5.4.1
4.8.27: Display of time/date stamp	The application must display to each user who retrieves the data the time and date on which the data were last modified.		Verify that the application displays to user who retrieves data time and date those data were last modified.	(V2)	BP	V1.1: 4.4.16, 5.4.2
4.8.28: Initialization of variables	The application code must explicitly initialize all of its variables when they are declared.	The programming language in which the application is written does not automatically ensure that all variables, when declared, are initialized to zero.	Verify that all application variables are initialized when declared.	(V23)	BP	V1.1: 4.4.18 Applications written in C will not automatically initialize declared variables to zero, as C does not provide this capability.
4.8.29: Hidden fields	The application must validate the source of all HTML updates to hidden fields and must reject any HTML field changes from unvalidated sources.	The application is a Web application, with Web pages that contain hidden fields	Verify that the application validates all sources of HTML field updates, and rejects all updates from unvalidated sources.	V21	BP	V1.1: 4.4.23

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.8.30: No parameter data in hidden fields	The application must not embed parameter data about fields in HTML forms in hidden fields in those HTML forms.	The application is a Web application that uses HTML forms	Verify that the application does not embed parameter data into hidden fields.	V2	BP	
4.8.31: Integrity of sensitive Web transactions	The application should use hash or digital signature to ensure the integrity of transmitted forms (user-to-server) containing sensitive information.	The application is a Web application:	Verify that the application uses cryptographic means to ensure the integrity of forms containing sensitive information. Validate that any cryptographic components within the application are approved for use.	V35	BP	
4.8.32: Distrust data received on untrustworthy channel	The application should not trust user input or other user-supplied data that have not been received over a trustworthy channel, unless those data are encrypted and digitally signed.		Verify that the application was designed to validate the source of all input before processing it. Verify requirement was designed into the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by conducting 3rd party source code review.	V32	BP	These data include (but are not limited to) cookies, hidden forms, email messages, and files/data containing reference/pointers to other files/data sent over untrustworthy channels.
4.8.33: Application response to untrustworthy input	The application must never return sensitive information in response to input from untrustworthy sources.		Verify that the application validates all input from untrustowrthy sources.	V2	BP	For even better security, the application should not respond at all to input from untrustworthy sources.
4.8.34: Reject Web page content from utrustworthy sources	The application must not accept Web page content from any untrustworthy source. The application must verify and validate the source of any Web page content before posting that content.	The application is a Web server application	Verify that the application does not store any sensitive information in any type of cookie or script. Inspect the contents of all scripts and cookies to determine if they contain sensitive data.	V1, V2, V22, V34	BP	
4.8.35: Integrity of electronic records	The application cannot be used to bypass the access controls or to spoof the trusted user to modify data within database entries/records (data integrity), the relationships between those entries/records (relational integrity), or the references to those entries/records (referential integrity)	The application is a database application	Verify that the application cannot be used by bypass the access controls providing data integrity, relational integrity, and referential integrity to the data in the database. Verify that the application cannot be used to enable a user to spoof a trusted user and modify the data records, relationships, or references.	V2	6 (C2.2.3.23)	
4.8.36: Resolution of mode changes	Before it shuts down, the application must reverse any changes in the application's operating mode or state that occurred during its execution, and must return to its normal mode and state of operation.		Verify that before shut down, the application reverts to normal mode of operation state.	(V29)	BP	V1.1: 4.4.17

4.11 Availability

This subsection lists requirements governing how applications ensure the availability of the data they manipulate, store, or transmit, as well as the availability of their own data, executables, and runtime resources. These requirements pertain to applications in environments in which the independent availability controls provided by the application's underlying host and surrounding security infrastructure are not considered adequate to protect the application and/or its data, and thus must be augmented by the application itself at the application layer. See IATF (Reference 27) Section 4.3.4 for a discussion of the processes and technologies involved in availability.

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.9.1: Application correctness guarantees data availability	The application code must not contain errors, bugs, or vulnerabilities that could cause any executing process within the application to inadvertently delete or overwrite data, to incorrectly assign/change access permissions to that data, or to otherwise impinge on the data's availability.		Verify that the application contains no vulnerabilities, errors, or bugs that cause application to overwrite data, misassign or modify access permissions to data, or otherwise affect data availability.	V36	BP	V1.1: 4.5.1, 5.5.1
4.9.2: Server code resistant to crash	The application code should not include bugs, errors, or exploitable vulnerabilities that could cause the executing application to crash.	The application is a server application.	Verify that application contains no vulnerabilities, errors, or bugs that cause application to crash.	V36	BP	V1.1: 4.5.2, 5.5.1
4.9.3: Server application resistant to DoS	The application code must not include bugs, errors, or exploitable vulnerabilities that could exploited by a malicious user or program to launch a successful DoS attack against the application.	The application is a server application	Verify, using a debugger and through execution testing, that the application code does not include bugs, errors, or exploitable vulnerabilities that could exploited by a malicious user or program to launch a successful DoS attack against the application.	V36	BP	V1.1: 4.5.2, 4.5.7, 5.5.1
4.9.4: Load level threshold	The application should enable the administrator to configure a load level threshold, and should stop processing incoming requests if that threshold is reached.	The application is a server application	Verify that the application enables the administrator to configure a load level threshold for its processing, and that it ceases to process requests when that threshold is reached.	V27	BP	
4.9.5: Process timeout	Every application process should be programmed with a defined threshold for the real time that can be used by that process. Once this timeout threshold is reached, the process should clean up all resources allocated to it by the host computer, and should terminate.		Verify that the application has defined thresholds for the real time that can be used by each of its component processes. Verify that when this timeout threshold is reached for a given process, the process clean ups all resources allocated to it by the host, and terminates.	V33	BP	V1.1: *4.2.10*
4.9.6: Adjust to unresponsive output	The application should be configured with a threshold whereby, if the application attempts to return data to a requesting client, but that client—or its network connection—does not respond after a certain period, the application will release its session locks and stop waiting for a client response.	The application is a server application	Verify that the application was designed to detect and resist denial of service attacks. Verify requirement was designed into the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by source code review and DoS testing.	V33	BP	
4.9.7: No interrupts between interdependent operations	The application should be coded or configured (in terms of its interaction with the underlying file system) in a way that prevents an interruption (i.e., to run an unrelated program) between two operations that are critically dependent on their sequential operation.		Verify that the application cannot be interrupted at a critical point in operation. Verify requirement was designed into the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by source code review and testing.	V3, V36	BP	
4.9.8: Client resistant to crash, MAC I	The application code must not include bugs, errors, or exploitable vulnerabilities that could cause the executing application to crash.	The application is a Mission Assurance Category 1 client application	Verify that the application contains no vulnerabilities, errors, or bugs that cause application to crash.	V36	BP	V1.1: 4.5.3 Requirement also applies to clients in other Mission Assurance Categories when the clients are considered high priority.
4.9.9: Client resistant to DoS, MAC I	The application code must not include bugs, errors, or exploitable vulnerabilities that could exploited by a malicious user or program to launch a successful DoS attack against the application.	The application is a Mission Assurance Category 1 client application	Verify that the application contains no vulnerabilities, errors, or bugs that make the application vulnerable to DoS attacks.	V36	BP	V1.1: 4.5.3, 4.5.7 Requirement also applies to clients in other Mission Assurance Categories when the clients are considered high priority.
4.9.10: Secure state after crash	When the application fails or is affected by an error condition, that failure/error must not cause the application to go into an insecure state.		Verify that an failure or error condition does not cause the application to go into an insecure state.	V15	2a (DCSS-2), 2b (DCSS-2), 2c (DCSS-2),	V1.1: 4.5.4

					2d (DCSS-2)	

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.9.11: Error/exception handling	The application must contain an error/exception handling capability that ensures that the application executable files and data will not become vulnerable in case of an application processing failure. The application must not rely on its programming language alone to perform error/exception handling.	The application is a server application	Verify that the application was designed with independent error handling provisions and that it does not solely rely inherent programming language error/exception handling. Verify requirement was designed into the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by source code review and testing. application.	V15	BP	
4.9.12: Error/exception handling resistant to DoS	The application's error handling and recovery capabilities must be robust enough that they cannot be overwhelmed by a flood of malformed arguments from malicious users or processes into a denial of service state; and the application's error-handling mechanism must be able to detect flood attacks and identify their source, and must be able to terminate processing related to any subsequent data or requests from the source of a detected flood attack.	The application is a server application	Verify that the application's error handling/recovery capability is robust enough to resist a denial of service attack involving a flood of malformed arguments.	V15	BP	V1.1: 4.5.8
4.9.13: Application failure notification	The application's error/exception handling capability must immediately notify the administrator by email, console message and/or pager message. The application must enable the administrator to configure which notification method(s) will be used.	An application process has failed.	Verify that the administrator is immediately notified when an application process fails.	V15	29 (3.2.4.1.1, 3.2.4.1.2)	V1.1: 4.5.5
4.9.14: Detection of external failures	The application must be able to detect failure conditions in the underlying host and surrounding infrastructure components with which it interfaces.		Verify that the application's error/exception handling capability is able to detect the failure of an infrastructure component with which it interfaces. Verify that the application's error/exception handling capability is able to detect the failure of an underlying host component with which it interfaces.	V15, V23	29 (3.2.4.1)	
4.9.15: Error/exception handling after infrastructure failure	The application must contain an error/exception handling capability that ensures that the application will avoid compromising the confidentiality, integrity, and availability of its executable files and data in case of a failure in one of the underlying host or infrastructure security mechanisms on which the application relies. The application's error handling capability should terminate the application in an orderly, secure manner when it detects a failure (lack of response) in one of the host or infrastructure security mechanisms on which the application relies.	The application is a server application	Verify that the application's error/exception handling capability terminates the application when one of its host or infrastructure security mechanisms fails.	V15	BP	

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.9.16: Configurable error/exception handling	The application's error handling mechanism must be configurable to allow the administrator to choose the way in which the application will respond to a detected error. At a minimum the options for error response should include: (1) Entire application terminates, (2) Erroneous process terminates, (3) Erroneous process and other selected processes terminate. In addition, the administrator should be able to choose one or more actions to be triggered by an error-related termination. These choices should include: (1) Termination triggers user notification, (2) Termination triggers administrator notification, (3) Termination triggers automatic checkpoint restart.		Verify that the application's error handling mechanism is configurable to determine error responses for various conditions (see box on left). Validate by generating error conditions which produce error handling output.	V15	BP	
4.9.17: Consistency check before restart	Before restart/recovery, the application must perform consistency checking to verify the validity of the application's call arguments, basic state assumptions, access control permissions and other security and critical parameters, data, and files.		Verify that the application ensures the integrity of all of its call arguments, parameters, state assumptions, data and files before it restarts.	V15	2a (COTR-1), 2b (COTR-1), 2c (COTR-1); 29 (3.2.4.2)	V1.1: 4.5.6
4.9.18: Trusted recovery	When the application is restarted, it must not cause the application to go into an insecure state.		Verify that a restart does not cause the application to go into an insecure state.	V15	2a (DCSS-2, COTR-1), 2b (DCSS-2, COTR-1), 2c (DCSS-2, COTR-1); 29 (3.2.4.1.4)	V1.1: 4.5.4
4.9.19: Checkpoint restart	The application must include a checkpoint restart capability that allows rollback after a transaction fails to the transaction's point in processing just before it failed.	The application is transaction-oriented.	Verify that upon detecting a transaction failure, the application roll back to the last validated transaction point. Verify requirement was designed into the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by source code review and testing.	V15	2a (ECDC-1), 2b (ECDC-1), 2c (ECDC-1);	
4.9.20: Limit error message data	Error messages returned by the application should report at most that a transaction/process has failed, with a minimal, generic description of the cause of the failure.		Verify that the application (in a production type environment) minimizes the content in error reporting messages. Verify requirement was designed into the application's development cycle. Post application development, verify that the requirement was properly implemented and functional by source code review and testing.	V15	BP	
4.9.21: Missing files	Before attempting to use any file or directory, the application must first verify that the file/directory exists on the system. If the file/directory is missing, the application must: (1) Return an error message informing the user that the requested file/directory cannot be found; (2) Gracefully terminate the user process through which the user requested the missing file/directory, and the server process that searched for that file/directory.		Verify that the application checks for existence of requested files and directories. Verify that the file/directory cannot be found, the application issues error message to the user warning that requested file/directory cannot be found, then gracefully terminates the user process and server process.	V29, V36	BP	V1.1: 4.5.9

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.9.22: Logging of failure events	The application's error/exception handling capability must log all error and failure events to an error/failure log.		Verify that the application's error/exception handling capability maintains a log file in which it logs information about all of the application's error events and failure events.	V15, (V24)	29 (3.2.4.1.3)	V1.1: 4.6.1
4.9.23: No core dumps	The application's error/exception handling capability must not cause the application program to perform a core dump, except during testing.		Verify that the application's error/exception handling mechanism is sufficiently capable and configurable to prevent a core dump from occurring in the event of a processing error.	V15	BP	

4.12 Accountability

This subsection lists requirements governing how applications ensure the accountability of users for the activities the perform and the application processes they spawn while using the application. These requirements pertain to applications for which user accountability via logging of application-specific events/transactions is required in addition to auditing at the operating system, Web server, and DBMS levels. See IATF (Reference 27) Section 4.3.5 for a discussion of the processes and technologies involved in accountability.

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.10.1: Audit/event logging mechanism	The application must log all security-relevant events (configured by the administrator) to its own secure audit/event log, or transmit these data securely to an external audit collection facility. In high-robustness applications, this audit mechanism must provide continuous, automated online auditing.		Verify that the application logs all security-relevant events either to its own secure audit file or to an external audit facility. Verify that, if the application is high-robustness, the audit mechanism provides continuous, automated online auditing.	V24	2a (ECAT-2), 2b (ECAT-2), 2c (ECAT-1). 2d (ECAT-2); 27 (7.1.4.7); 29 (3.2.17.5)	V1.1: 4.6.1
4.10.2: Configurable audit parameters	The audit facility used by the application must allow the administrator to select the events to be logged and the information to be captured about each event.		Verify that the audit mechanism used by the application allows the administrator to select types of events to be logged and type information to capture about each event.	V24	29 (3.2.3.2, 3.2.3.4, 3.2.3.8, 3.2.16.7)	V1.1: 4.6.2
4.10.3: Events to be audited	The application must log the following types of events to its audit facility, at a minimum: (1) Startup and shutdown, (2) Authentication, (3) Authorization/permission granting, (4) Actions by trusted users, (5) Process invocation, (6) Controlled access to data by individually authenticated user, (7) Unsuccessful data access attempt, (8) Data update, (9) Data deletion, (10) Input validation, (11) Establishment of network connection, (12) Data transfer, (13) Application configuration change, (14) Application of confidentiality or integrity labels to data, (15) Override or modification of data labels or markings, (16) Output to removable media, (17) Output to a printer, (18) For classified applications: Changes of sensitivity labels on application-accessed data objects.		Verify that the application logs the specified list of events of its audit facility.	V24	2a (ECCD-2), 2b (ECCD-2), 2c (ECCD-1), 2d (ECCD-1, ECLC-1); 29 (3.2.3.3, 3.2.6.4.1, 3.2.17.5)	V1.1: 4.6.3

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.10.4: Binding UserID to audit record	The audit facility used by the application must bind the individual ID of the user causing (or associated with) the audited event to the audit record for that event.		Verify that the audit facility binds the UserID to the audit record.	V24	29 (3.2.1.3, 3.2.3.5)	V1.1: 4.6.4
4.10.5: Audit data, classified or MAC I application	Each audit record must include the following information (as relevant for the type of event): (1) UserID of user or process ID of process causing the event, (2) Successful or failure of attempt to access a security file, (3) Date and time of the event, (4) Type of event, (5) Success or failure of event, (6) Seriousness of event (violation, 7) Successful or failure of login attempt, (8) Denial of access resulting from excessive number of login attempts, (9) Blocking or blacklisting a UserID, terminal, or access port, and the reason for the action, (10) Data required to audit the possible use of covert channel mechanisms, (11) Privileged activities and other system level access, (12) Starting and ending time for access to the application, (13) Activities that might modify, bypass, or negate safeguards controlled by the system, (14) Security-relevant actions associated with periods processing, or the changing of security labels or categories of information, (15) For I&A events: origin of request (e.g., originating host's IP address, 16) For write or delete events: name of data object written or deleted.	The application handles classified data or is a MAC I application.	Verify that application audit records contained the specified information as indicated.	V24	2d (ECAR-3); 29 (3.2.3.5, 3.2.3.6, 3.2.3.7)	V1.1: 4.6.5
4.10.6: Audit data, sensitive, private, or MAC II application	Each audit record must include the following information (as relevant for the type of event): (1) UserID of user or process ID of process causing the event, (2) Success or failure of attempt to access security file, (3) Date/time of event, (4) Type of event, (5) Success or failure of event, (6) Seriousness of event (violation, 7) Success or failure of login attempt, (8) Denial of access resulting from excessive number of login attempts, (9) Blocking or blacklisting of UserID, terminal, or access port, and reason for the action, (10) Activities that might modify, bypass, or negate security safeguards controlled by the application, (11) For I&A events: origin of request (e.g., originating host's IP address), (12) For write or delete events: name of data object written or deleted	The application handles sensitive or unclassified data that is not publicly releasable, or is a MAC II application.	Verify that application audit records contained the specified information as indicated.	V24	2e (ECAR-2); 29 (3.2.3.5, 3.2.3.6, 3.2.3.7)	V1.1: 4.6.6
4.10.7: Audit data, public access or MAC III application	Each audit record must include the following information (as relevant for the type of event): (1) UserID of user or process ID of process causing the event, (1) Success or failure of attempt to access security file, (2) Date/time of event, (3) Type of event, (4) Success or failure of event, (5) Seriousness of event (violation), (6) For I&A events: origin of request (e.g., originating host's IP address), (7) For write or delete events: name of data object written or deleted.	The application handles publicly releasable data only or is a MAC III application.	Verify that application audit records contained the specified information as indicated.	V24	2f (ECAR-1); 29 (3.2.3.5, 3.2.3.6, 3.2.3.7)	V1.1: 4.6.7

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.10.8: Audit of schema objects	The application's audit facility shall log all schema objects, with auditing able to be turned on or off on a per-object basis.	The application is database or directory application	Verify that all schema objects are being audited (e.g., by querying the DBA_OBJ_AUDIT_OPTS table in Oracle).	V24	BP	V1.1: 5.6.1
4.10.9: Audit trail "fill thresholds"	The audit facility used by the application shall enable the administrator to set the audit trail "fill thresholds" as follows: (1) A threshold that indicates the audit trail is some percentage full, which shall trigger a notification to the administrator that the file should be archived and purged; (2) A threshold that indicates the audit/log file is full, which shall trigger one of the following events (configurable by the administrator): graceful shutdown of the application, OR suspension of user processing, OR overwriting of the oldest audit records, OR termination of auditing, OR increase of storage space allotted for audit records (to an amount configurable by the administrator).		Verify that the audit facility application ensures that the application's log records are protected from unauthorized deletion, disclosure, or modification.	V24	29 (3.2.3.1.3)	V1.1: 4.6.9
4.10.10: Fill threshold notification	The audit facility used by the application shall notify the administrator when the audit trail's fill threshold is being approached. The administrator shall be able to configure the percentage full at which the audit trail must be for this notification to be triggered.		Verify that the audit facility enables the administrator to configure "log fill" thresholds and "audit full" events as indicated. Verify that administrator is notified by the audit facility when the "almost full" threshold is reached. Verify that the event configured by the administrator is triggered when the "log full" threshold is reached.	V24	29 (3.2.3.1.3)	V1.1: 4.6.9
4.10.11: Security violation notifications, MAC I application	The audit facility used by the application must: (1) Immediately alert the security administrator of all security violations and unusual or suspicious activity that might indicate a security violation. (2) Enable the administrator to configure the audit facility to automatically shut down the application if a detected security violation is considered serious enough to warrant it.	Application is a MAC I application.	(1) Verify that the administrator is immediately notified when a security violation is detected. (2) Verify that the administrator can configure the violation(s) considered serious enough to warrant automatically shutting down the application. (3) Verify that the application does shut down when an administrator-selected serious violation is detected.	V24	2a (ECAT-2), 2d (ECAT-2)	V1.1: 4.6.11 Serious events constitute suspicious, unusual, or inappropriate activities that indicate possible serious security violations and warrants shutting down the application to prevent escalation of risk. The administrator should assign each event a "seriousness" rating when configuring auditing for the application.
4.10.12: Audit viewing and reporting tool	The audit facility used by the application shall include a tool that enables the administrator to view the application's audit records, and to report against them.		Verify that the application audit/event logging mechanism immediately alerts the administrator when it detects an actual or potential security violation. Verify that audit/log mechanism triggers a graceful shutdown of application if event is "serious".	V24	2a (ECRG-1), 2b (ECRG-1), 2c (ECRG-1); 29 (3.2.3.1.5, 3.2.3.9, 3.2.3.10, 3.2.3.11)	V1.1: 4.6.12
4.10.13: Audit failure	The application must notify the administrator and, as configured by the administrator, either: (1) Shutdown the application, OR (2) Suspend user processing, OR (3) Initiate an automatic restart of the audit facility	The application audit facility has failed	Verify that the audit facility provides a tool for viewing and reporting against application audit records.	V15	29 (3.2.3.1.4)	V1.1: 4.6.10
4.10.14: Integrity and availability of audit records	The audit facility used by the application shall ensure that the application's audit records are protected from deletion or unauthorized modification.		Verify that the application logs all security-relevant events either to its own secure audit file or to an external audit facility.	V2	2a (ECTP-1), 2b (ECTP-1), 2c (ECTP-1); 29 (3.2.3.1,	V1.1: 4.6.8

					3.2.16.8)	

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.10.15: Confidentiality of audit records	The audit facility used by the application shall ensure that the application's audit records are protected from unauthorized disclosure.		Verify that the audit mechanism used by the application allows the administrator to select types of events to be logged and type information to capture about each event.	V2	2a (ECTP-1), 2b (ECTP-1), 2c (ECTP-1); 29 (3.2.3.1, 3.2.16.8)	V1.1: 4.6.8
4.10.16: Access control of audit data and processes	The access controls used by the application shall prohibit read, write, delete, execute, move, or copy access to the application's audit records and processes by unauthorized users.		Verify that the application's access control mechanisms prevent unauthorized users from gaining read, write, execute, delete, move, or copy access to any of the application's audit files or processes.	V2	2a (ECTP-1), 2b (ECTP-1), 2c (ECTP-1); 29 (3.2.3.1.1, 3.2.3.1.2, 3.2.16.8)	V1.1: *4.6.8*

4.13 Non-Repudiation

This subsection lists requirements governing how applications ensure non-repudiation by users of activities they perform, processes they spawn, and data they create, modify, delete, or transmit while using the application. These requirements are pertinent for applications whose users must be held accountable for individual transactions, particularly those transactions performed using different application components over a network (e.g., email user agent and email server, browser and Web server, directory user agent and directory server, etc.) for which traditional auditing and logging would not be sufficient to maintain and easily track user accountability, as well as those applications for which non-repudiation is a legal requirement. See IATF (Reference 27) Section 4.3.5 for a discussion of the processes and technologies involved in non-repudiation.

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.11.1: Proof of transmission	The application must invoke a NIST- or NSA-approved digital signature technology (e.g., SHA-1, DSA, RSA) appropriate to the application's Mission Assurance Category to enable the creator/sender to digitally sign the data/keys the application is used to create or transmit overThe network.	The application requires non-repudiation by the creator or sender of data or cryptokeys transmitted via yhe application.	Verify that the application invokes appropriate approved digital signature technology to enable users to digitally sign data/cryptokeys prior to transmission.	V24	2a (DCNR-1), 2b (DCNR-1), 2c (DCNR-1); 27 (4.5.3.8, 7.1.4.6); 29 (3.2.14.1)	V1.1: 4.7.1
4.11.2: Proof of delivery	The application must invoke a NIST- or NSA-approved digital signature technology (e.g., DSS) appropriate to the application's Mission Assurance Category to enable recipient to sign data received viaThe application.	The application requires non-repudiation by the recipient of data received viaThe application.	Verify that the application invokes appropriate approved digital signature technology to enable users to digitally sign data upon receipt.	V24	27 (7.1.4.6); 29 (3.2.14.2)	V1.1: 4.7.2
4.11.3: Digital signature validation	The application must invoke a digital signature validation facility to validate all digital signatures applied to data or cryptokeys it receives over the network or retrieves from a database or directory.		Verify that receiving application validates digital signatures on data/cryptokeys it receives over a network.	V24	27 (7.1.4.6)	V1.1: 4.7.3
4.11.4: Protection of signature security data	The application must protect from tampering with and inappropriate disclosure of the cryptokeys and certificates it uses for digital signature processing.		Verify that the application's digital signature-related cryptokeys and certificates are adequately protected from tampering and inappropriate disclosure.	V2	BP	V1.1: 4.7.4

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.11.5: Digital signature of email messages	The application must invoke a PKI for digital signature of messages using Class 3 identity certificates, with a transition to Class 4 identity certificates requiring minimal modification to the application code byThe deadline specified in DOD PKI Policy.	The application is a messaging or email application.	Verify that email application supports digital signature using Class 3 certificates and can accommodate use of Class 4 with minimal modification to application code.	(V23)	3 (Digitally Signed Email); 29 (3.2.21.3, 3.2.21.9)	V1.1: *4.7.5*
4.11.6: Digital signature of sensitive Web input	The application should use digital signature to ensure the non-repudiation of transmitted forms (user-to-server) containing sensitive information.	The application is a Web application	For input deemed sensitive, verify that the application uses digital signatures on transmitted forms for purposes of non-repudiation. If digital signature is invalid, input should be rejected.	V24	BP	V1.1: *4.7.1*

4.14 Preparation for Deployment

This subsection lists requirements governing how the application the application is prepared for deployment—e.g., "clean up" of application code, installation and configuration in anticipation of operation, etc. These requirements apply to all applications (within the constraints defined in the Assumptions and Constraints for a given requirement), regardless of what security functions they perform.

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.12.1: Cleanup for deployment	Remove any residual backup files, temporary files, File Transfer Protocol (FTP) programs, and debugging files, tools, accounts, passwords, debug and test flags, and other unnecessary files and developer "backdoors" from the application code and its underlying host environment.		Verify that the application code and the platform on which it runs contain no unnecessary files or developer "backdoors" of any kind.	V17, V36	BP	V1.1: 4.0.11
4.12.2: Remove debug options	Do not deploy code that has been compiled with debugging options.		Verify that the final, production version of the deployed application was compiled with debug options disabled.	V17, V36	BP	
4.12.4: Remove source code comments	Remove all references and comments from HTML/source code that reveal features of the application's design, underlying Web server or file system directory structure. Such information includes (but is not limited to): (1) Directory structures, (2) Location of the Web root, (3) Debug information, (4) Cookie structures, (5) Problems associated with development, (6) Developers' names, email addresses, phone numbers.	The application contains HTML code or other source code that can be displayed by end-users.	Verify that HTML or other source code contains no comments that include information that could be exploited by an attacker to attack the application or its host environment.	V29, V36	BP	V1.1: 4.0.13
4.12.5: Automatically generated HTML tags	Remove all non-standard, erroneous (e.g., syntax errors), and unnecessary tags from HTML code. Also remove all tags that are not browser neutral (i.e., that are intended to optimize the code for a particular browser).	The application contains HTML that was automatically generated y a Web authoring tool	Verify that the HTML code contains no tags that serve no obvious purpose, or that cause the Web page to: () display in an unexpected way; () crash, "freeze", or otherwise detrimentally affect the operation of the browser; () unintentionally jiggle, flash, wobble, etc; () display differently in browsers from different vendors.	V29	BP	

Requirement	*Description*	*Assumptions and Constraints*	*Test Objective*	*Vulnerability Addressed*	*Policy Source*	*Note*
4.12.6: Default accounts	Disable all default accounts if not absolutely necessary. Change passwords on any default accounts that remain.	The application includes COTS components	Verify that the application (and platform that contains it) has had all default accounts disabled. Verify that the platform has undergone the STIG process.	V29	2a (ECSC-1), 2b (ECSC-1), 2c (ECSC-1), 2d (IAIA-2), 2e (IAIA-2)	
4.12.7: Unnecessary calls	Remove any calls from the application code that do not accomplish anything.		Verify that any unnecessary calls from the application have been removed from the application code. Verify by source code review and testing.	V25, V36	BP	Examples of unnecessary calls are calls to external processes or libraries that do not exist, have been replaced.
4.12.8: Access control configuration	Access controls shall be configured to enforce access restrictions in accordance with Paragraph C2.2.7 Table C2. T5 of ASD C3I Memorandum, "Policy Guidance for Use of Mobile Code Technologies in Department of Defense (DOD) Information Systems", 7 November 2000.	The application is an electronic records management application	Verify that the access controls relied upon by the application are configured in accordance with the referenced document.	V2	6 (C.2.2.7)	
4.12.9: STIG-compliant configuration	The installed application's runtime environment (underlying Web server, database management system, operating system, infrastructure components, etc.) must be configured in compliance with all relevant STIGs. If there is a STIG that is relevant for the application itself, the application must be configured according to this STIG.		Verify that the underlying host, and all middleware, infrastructure, and related components that comprise the application's runtime environment are configured in accordance with all relevant STIGs. Verify that the application itself is configured in accordance with the relevant application STIG, if one exists.	V29	1 (4.18); 2a (ECSC-1), 2b (ECSC-1), 2c (ECSC-1)	
4.12.10: Assignment of privileges	Trusted accounts/roles (e.g., Administrator) must be assigned privileges to access trusted functions only, and not to any non-trusted functions. Non-trusted accounts/roles (e.g., User) must be assigned privileges to access non-trusted functions only, and not to any trusted functions.		Verify that accounts for administrators and other trusted user roles have been granted privileges to perform the trusted functions associated with those roles, and are not allowed to perform any non-trusted (end-user) functions. Verify that accounts for end users and other non-trusted roles have been granted privileges to perform only non-trusted functions.	V26		V1.1: *5.2.1*, *5.2.2*, *5.2.5*, *5.2.6*, *5.2.7*
4.12.11: PBAC or RBAC configuration	The application's access controls—DAC and MAC—must be configured so that they grant or deny access to users and processes based on the privileges authorized to those users/processes under the application's PBAC or RBAC scheme.		Verify that the application's access controls are configured to be consistent with the role-associated privileges assigned to different roles under the RBAC scheme implemented by the authorization mechanism used by the application.	V2, (V26);	2a (ECPA-1), 2b (ECPA-1), 2c (ECPA-1)	An example: DACs on the application's configuration files should be configured to grant write-access to all users in the Administrator role, read-access to Process role, and no access to any other role.
4.12.12: No "nobody" account	The Web server's "nobody" account should be disabled, and all programs and scripts that are intended to run as the Web server's "nobody" user should be modified to run under a specific username.	The application is a Web server application.	Verify that the server has been configured are in accordance with Security Technical Implementation Guides (STIGs). Specifically, verify that the web server's "nobody" account has been disabled, if applicable.	V29	BP	
4.12.13: System library access controls	Access controls on all system libraries accessed by the application must be configured to protect the application's privileged programs, and to prevent introduction of unauthorized code.	The application is a server application.	Verify that the server's access controls have been configured to prevent unauthorized access to the application's privileged programs, and to prevent introduction of code by unauthorized users.		2a (DCSL-1), 2b (DCSL-1), 2c (DCSL-1)	
4.12.14: Host access controls	Access controls in the host infrastructure that provides the application access to any security infrastructure components must isolate through partitioning all infrastructure security components' processes invoked by the application, as well as the application's security-related processes, in separate execution domains that are separate from non-security processes.		Verify that the server's access controls partition all security component processes and application processes that use those security component processes into separate execution domains that are separate from non-security processes.		2a (DCSP-1), 2b (DCSP-1)	

(Page intentionally blank)

APPENDIX A: ACRONYMS AND ABBREVIATIONS

ACL: Access Control List

AES: Advanced Encryption Standard

AFS: Andrew File System

ANSI: American National Standards Institute

API: Application Programmatic Interface

ASD C3I: Assistant Secretary of Defense for Command, Control, Communications & Intelligence

C&A: Certification and Accreditation

CAC: Common Access Card

CASE: Computer Aided Software Engineering

CIO: Chief Information Officer

CERT: Computer Emergency Response Team

CGI: Common Gateway Interface

CJCS: Chairman of the Joint Chiefs of Staff

CINC: Commander-in-Chief

COE: Common Operating Environment

CORBA: Common Object Request Broker Architecture

COTS: Commercial-off-the-Shelf

CRL: Certificate Revocation List

CPU: Central Processing Unit

CSL: Computer Systems Laboratory

DAC: Discretionary Access Control

DBA: Database Administration

DBMS: Database Management System

DCOM: Distributed Component Object Model

DES: Data Encryption Standard

DID : Defense in Depth

DII: Defense Information Infrastructure

DISA: Defense Information Systems Agency

DISAI: Defense Information Systems Agency Instruction

DITSCAP: Department of Defense Information Technology Security Certification and Accreditation Process

DGSA: Department of Defense Goal Security Architecture

DoS: Denial of Service

DOS: Disk Operating System

DOD: Department Of Defense

DODD: Department of Defense Directive

DSD: Deputy Secretary of Defense

EAL: Evaluation Assurance Level

E-mail: Electronic Mail

FTP: File Transfer Protocol

FIPS: Federal Information Processing Standard

GENSER: General Service

GIG: Global Information Grid

GUI: Graphical User Interface

HTTP: Hypertext Transfer Protocol

HTML: Hypertext Markup Language

I&A: Identification and Authentication

IA: Information Assurance

IATF: Information Assurance Technical Framework

IAVA: Information Assurance Vulnerability Alert

ID: Identification

IIS: Internet Information Server

IPC: Interprocess Communication

ISS: Internet Security Systems

KRL: Key Revocation List

MAC: Mandatory Access Control

MAC I: Mission Assurance Category I

MAC II: Mission Assurance Category II

MAC III: Mission Assurance Category III

MS: Microsoft

NFS: Network File System

NIAP: National Information Assurance Partnership

NIST: National Institute of Standards and Technology

NSA: National Security Agency

OPSEC: Operations Security

OS: Operating System

OSD: Office of the Secretary of Defense

OWASP: Open Web Application Security Project

PDF: Portable Document Format

PHP: PHP Hypertext Preprocessor

PKE: Public Key Enabling

PKI: Public Key Infrastructure

QA: Quality Assurance

RBAC: Role-Based Access Control

SAMI: Sources and Methods Intelligence

SBU: Sensitive But Unclassified

SETUID: Set User Identification

SETGID: Set Global Identification

SFDG: Security Features Developers Guide

SGA: System Global Area

SML: Strength of Mechanism Level

SNAC: Systems and Network Attack Center

SNMP: Simple Network Management Protocol

SP: Special Publication

SQL: Structured Query Language

SRS: Software Requirements Specification

SSI: Server Side Include

SSL: Secure Sockets Layer

SSO: Single Sign-on

STD: Standard

ST&E: Security Test and Evaluation

STIG: Security Technical Implementation Guide

SYS: System

SYSDBA: System Database Administrator

SYSOPER: System Operator

TAFIM: Technical Architecture Framework for Information Management

TCB: Trusted Computing Base

TS/SCI: Top Secret/Sensitive Compartmented Information

URL: Uniform Resource Locator

VBA: Visual Basic for Applications

VPN: Virtual Private Network

WSH: Windows Scripting Host

APPENDIX B: REFERENCES

This section lists policies, instructions, guidance, and application working groups that were used as sources when specifying the application security requirements in this document. The section also lists recognized sources of best practices for application and software security consulted when developing this document.

B.1 DOD-Wide Policy and Guidance

1. Department of Defense Directive (DODD) 8500.1, "Information Assurance (IA)" (24 October 2002)

2. DOD Instruction (DODI) 8500.2, "Information Assurance (IA) Implementation" (6 February 2003), and specifically the following attachments:

   a. E4.A1. Attachment 1 to Enclosure 4: Mission Assurance Category I Controls for Integrity and Availability

   b. E4.A2. Attachment 2 to Enclosure 4, Mission Assurance Category II Controls for Integrity and Availability

   c. E4.A3. Attachment 3 to Enclosure 4: Mission Assurance Category III Controls for Integrity and Availability

   d. E4.A4. Attachment 4 to Enclosure 4: Confidentiality Controls for DOD Information Systems Processing Classified Information

   e. E4.A5. Attachment 5 to Enclosure 4: Confidentiality Controls for DOD Information Systems Processing Sensitive Information

   f. E4.A6. Attachment 6 to Enclosure 4: Confidentiality Controls for DOD Information Systems Processing Publicly Released Information

3. ASD C3I Memorandum, "Department of Defense (DOD) Public Key Infrastructure (PKI)", 12 August 2000 (known as DOD PKI Policy).

4. ASD C3I Memorandum, "Public Key Enabling (PKE) of Applications, Web Servers, and Networks for the Department of Defense (DOD)", 17 May 2001 (known as DOD PKE Policy).

5. ASD C3I Memorandum, "Policy Guidance for Use of Mobile Code Technologies in Department of Defense (DOD) Information Systems", 7 November 2000 (known as DOD Mobile Code Policy), specifically Enclosure 1.

6. DOD 5015.2-STD, Design Criteria Standard for Electronic Records Management Software Applications, (19 June 2002).

7. Chairman of the Joint Chiefs of Staff (CJCS) S3231.01, Safeguarding the Single Integrated Operational Plan (U), 30 November 1993.

8. DOD Technical Architecture Framework for Information Management (TAFIM), volume 6, DOD Goal Security Architecture (DGSA), 30 April 1996.

9. DOD Web Site Administration Policies and Procedures, 25 November 1998.

10. DOD Computer Emergency Response Team (CERT) Information Assurance Vulnerability Alerts (IAVA).

11. Deputy Secretary of Defense (DSD) Memorandum, Web Site Administration, 7 December 1998.

12. X.509 Certificate Policy for the United States Department of Defense, Version 6.0 (31 May 2002).

13. Department of Defense Target Public Key Infrastructure Operational Requirements Document (20 August 2001), specifically Section 1.4.1.2, "PK-Enabled Application Operation".

14. Department of Defense Directive 85xx.xx (DRAFT), "Biometric Technologies", Version 6.3.51 (undated). NOTE: This draft policy is based on ASD C3I Memorandum on Biometrics dated 19 January 2001, and Deputy Secretary of Defense Memorandum on Biometrics dated 27 December 2000.

15. Office of the Secretary of Defense (OSD) Memo, "Privacy Polices and Data Collection on DOD Public Web Sites" (13 July 2000), which is based on and in compliance with OMB Director Jacob J. Lew's Memorandum to the Heads of Executive Departments and Agencies (M-00-13, 22 June 2000).

16. Department of Defense (DOD) Class 3 Public Key Infrastructure (PKI) Public Key-Enabled Application Requirements (13 July 2000).

17. ASD C3I Memorandum, "Public Key Infrastructure (PKI) Policy Update" (21 May 2002).

18. ASD(C3I) Action Memo, "Updated Guidance for Public Key Infrastructure Policy Milestones" (4 February 2002).

B.2 DISA Policy and Guidance

19. DISA Defense Information Infrastructure (DII) Common Operating Environment (COE) Security Software Requirements Specification (SRS), Version 4, 20 October 1998.

20. DISA DII COE/GCCS/GCSS Application Security Requirements and Assessment Guidance Document (draft), 22 March 2002.

21. DII COE UNIX Application and Kernel Developer's Security Guidance.

22. DII COE Windows NT Application and Kernel Developer's Security Guidance.

23. DII COE 4.2.0.0 Security Features Developers Guides (SFDG).

24. DISA Instruction (DISAI) 630-230-19, Information Systems Security Program, July 1996.

25. DISA Web Application Security Technical Implementation Guide (STIG), Version 2, Release 2, 27 February 2001.

26. DISA Database Security Technical Implementation Guide (STIG), 28 September 2001.

B.3 Intelligence Community Policy and Guidance

27. NSA Information Assurance Technology Framework, Release 3.1, September 2002.

28. NSA Network Applications Team of the Systems and Network Attack Center (SNAC): Guides to the Secure Configuration and Administration of COTS Servers.

29. Defense Intelligence Agency: DII COE Security Requirements Specification (SRS), Version 4 (20 October 1998)

B.4 Civilian Agency Policy and Guidance

30. National Institute of Standards and Technology (NIST) Special Publication (SP) 800-26, Security Self-Assessment Guide for Information Technology Systems, November 2001. Web Reference: http://csrc.nist.gov/publications/nistpubs/800-26/sp800-26.pdf

31. NIST Special Publication (SP) 800-44, Guidelines on Securing Public Web Servers, February 2002.

B.5 Best Practices

32. Open Web Application Security Project (OWASP): "The Ten Most Critical Web Application Security Vulnerabilities" (13 January 2003).

33. David Scott and Richard Sharp, University of Cambridge: "Developing Secure Web Applications", in IEEE Computer Society, *Securing Your Systems for 2003 and Beyond* (November-December 2002).

34. Joseph Yoder and Jeffrey Barcalow: "Architectural Patterns for Enabling Application Security" (1998).

35. Internet Security Systems, Inc.: *Database Scanner, Version 4.2.0*.

36. Razvan Peteanu: "Best Practices for Secure Development", Version 4.03 (October 2001).

37. Andrew Jaquith, @Stake Research Report: "The Security of Applications: Not all Are Created Equal" (February 2002).